CMD

RAMLink™

USERS MANUAL

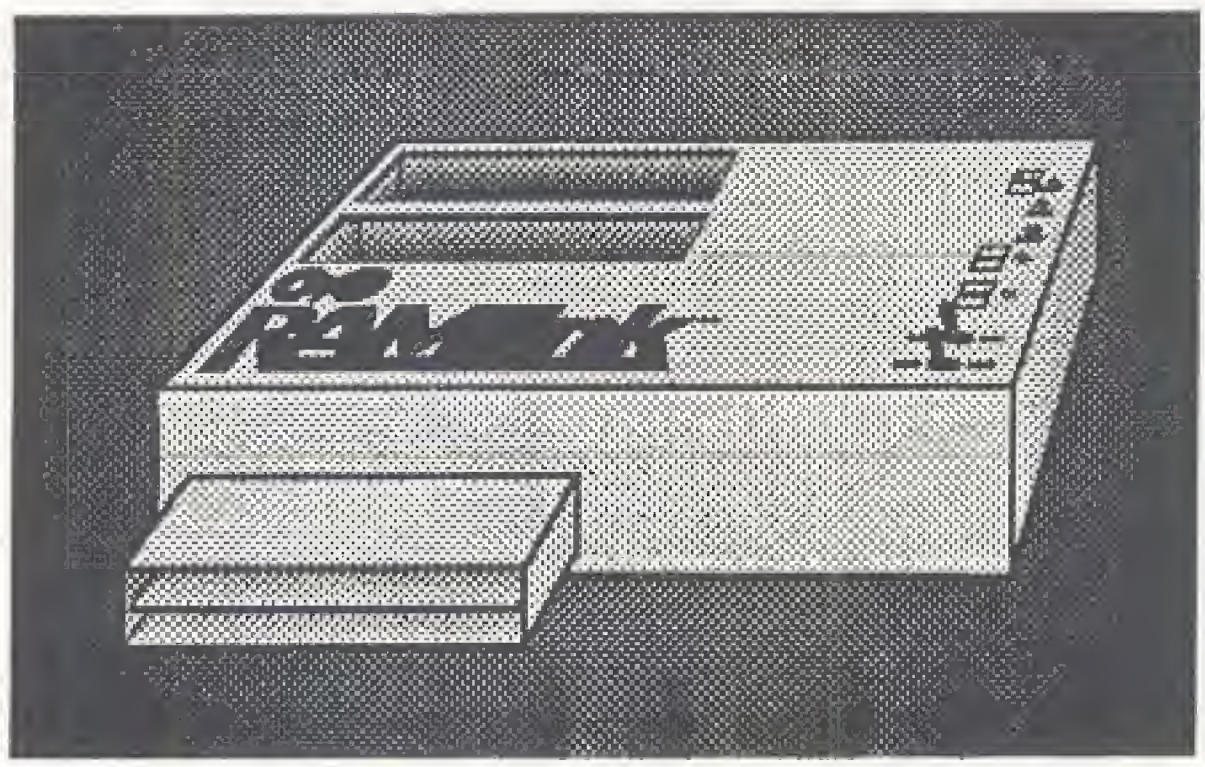


Table of Contents

Section 1: General Information

Section 2: Installing RAMLink

Section 3: Special Features

Section 4: Partitions and Subdirectories

Section 5: JiffyDOS

Section 6: Using Software

Section 7: Command Reference

Appendices

Appendix A: Utilities

Appendix B: Error Codes

Appendix C: Partition and File Formats

Appendix D: RAMLink Memory Map

Appendix E: Installing RAMCard

Appendix F: Installing SIMMs

Appendix G: Parallel Port

Appendix H: Power Connector

Appendix I: Battery Connector

Warranty Information

Section 1
General Information

Introduction

RAMLink extends the capabilities and compatibility of RAM expansion units for the Commodore 64 and 128 series of computers. You may also use RAMLink as a compatible RAM disk by installing RAM internally. Other benefits include built-in JiffyDOS Kernal routines, a parallel hard drive interface for the CMD HD series, an external power supply, and an optional battery backup unit. Compatibility with the widest possible range of software products has been incorporated plus many special features and the capability to expand storage space used by the system.

Main Functions

RAMLink, while mainly designed as a compatibility interface for RAM, also provides several other functions for users of Commodore 64 and 128 computer systems. For this reason, we refer to RAMLink as a multi-function compatibility interface. The main functions provided by RAMLink are:

- RAM COMPATIBILITY INTERFACE - Provides a usable DOS interface for Commodore 1700, 1764 and 1750 RAM Expansion Units and for Berkeley Softworks' GEORAM.
- RAM EXPANDER - Internal RAM capacity allows using RAMLink as a stand-alone REU.
- RAM DATA RETENTION - Separate power supply and optional battery backup allows RAMLink to retain data and programs when the host computer is off.
- JIFFYDOS - Built-in JiffyDOS Kernal routines (equivalent to the computer portion of JiffyDOS) for both the Commodore 64 and 128.
- HIGH SPEED PARALLEL PORT - Provides the fastest data transfer rates possible with the CMD HD Series hard drives.
- CARTRIDGE PORT EXPANDER - By providing a PASS-THRU port, RAMLink allows you to use RAM expansion concurrently with popular utility and I/O cartridges for the Commodore 64 and 128.

About RAM Disks

Many popular computer systems support creating and using RAM disks. Often, this is accomplished by using part of the internal RAM, such as on Amiga and IBM computers. Because of the limited amount of RAM within the Commodore 64 and 128, it is preferable to add external RAM for a RAM disk. A few companies have produced such RAM expansion units, yet most of these have been ineffective, suffering from incompatibilities with popular software. Two of the main reasons that RAM expanders have not delivered the compatibility necessary for most software are that they have lacked a proper DOS emulation and have not been transparent to the system. RAMLInk overcomes these shortcomings by providing a transparent DOS interface for access to the RAM disk.

Accessories

Several accessories are available from CMD which will extend the capabilities of your new RAMLink system. The following is a list of these items, and some benefits of using them. Prices, if given, are retail and do not include shipping or any applicable tax. Prices, specifications and availability are subject to change without notice. For further information on these products, contact CMD.

RAMCard

RAMLink contains an internal connector for adding RAMCard, which is a 'daughter board' that allows you to easily expand the RAM capacity of RAMLink up to a maximum of 16 Megabytes. RAMLink may contain only one RAMCard, which in turn may hold up to four SIMM modules mounted in sockets on RAMCard. The SIMM modules may be 1 or 4 Megabyte versions, but mixing of these two types is not allowed. When using RAMLink with an internal RAMCard, the memory found on the card may be either combined with the memory found in an external RAM expander (Commodore REU, Berkeley GEORAM or CMD RAMDrive√), or these may be utilized as separate entities.

SIMMs

RAMCard comes supplied with up to four SIMMs (memory modules). If your RAMLink is fitted with a RAMCard which contains less than four SIMMs, you may add more SIMMs to expand the available amount of RAM storage space. Two types of SIMMs are available - 1 Mb by 8 or 4 Mb by 8 (100 ns or faster). Mixing of the two types is not allowed, so if you wish to expand beyond 4 Megabytes you may only use the higher capacity SIMMs, and any lower capacity SIMMs must be removed.

JiffyDOS Drive ROMs

The computer portions of JiffyDOS for both the 64 and 128 are contained within RAMLink's RL DOS. By adding JiffyDOS drive ROMs to the disk drives on your system, you will be able to speed up the transfer rate of those drives by as much as 15 times. Drive ROM installation is simple and quick, and we supply complete installation instructions for every drive type supported. CMD supplies ROMs for eighteen different models of Commodore or compatible serial bus disk drives.

Battery Backup

RAMLink provides a connector for attaching an optional battery. This is used to prevent the loss of RAM contents if a power failure should happen, or while transporting RAMLink to another computer or location. The battery will provide short-term backup from a few hours to a day or so, and the length of backup time is dependant upon the amount of RAM installed. Battery power will protect any RAM installed on a RAMCard and any RAM expander attached via the RAM port.

Parallel HD Cable

RAMLink provides a method to connect the CMD HD Series hard drives to your computer via the hard drive's parallel connector. This speeds up disk access time substantially with the HD, providing transfer rates of approximately 51 K bytes per second in 64 mode, and 100 K bytes per second in 128 mode.

Orders and Information

If you wish to place an order or need general information about any product available from us, you should call Monday through Friday 10:00 am through 5:00 pm ET. If possible, call before 3:00 pm ET for fastest service. The number to call is (413) 525-0023.

You may also call this number during these hours if you have a problem with an order you placed with CMD. If you have a problem with an order which you placed with one of our dealers, you should contact the dealership.

If You Need Technical Assistance

Several forms of technical assistance are available to CMD product owners. You may contact us by modem, telephone, or mail. If you experience a problem which requires immediate assistance, contact us at the number above. Whenever calling, have your serial number, model number, and any other important information ready. Jot down this information on a copy of the Problem Reporting Log found elsewhere in this manual.

If you have programming questions, or need non-immediate technical assistance, the best method to use is one of the two telecommunications services where support areas for CMD customers are available. The first of these is Q-Link, the most widely used service for Commodore computer users. Our support section is in the 'Commodore Information Network' (CIN) area of Q-Link. After entering this area, pick 'Hardware Support Groups'. At the next menu, selecting 'Hardware Company Support Area' will present you with a menu containing the name of our company, 'Creative Micro Designs'. Pick this choice and you will be in our area. Q-Link is online from 6:00 pm until 7:00 am ET (daily) and 6:00 pm Friday through 7:00 am Monday ET (weekends). We highly recommend this service to Commodore users.

CMD also maintains its own telecommunication service. This is available 24 hours a day at 300/1200/2400 baud. The communications parameters are 8 data bits, no parity, and 1 stop bit (8-N-1 or 8-None-1). Latest updates of CMD utilities are available on this system. This system also provides support files for BBS programs and general discussion areas. The telephone number for this service is (413) 525-0148.

Questions left to us on either of these services are usually answered within 48 hours. If you are unable to use one of these services to contact us about technical matters, contact us at our voice telephone number given earlier.

Bug reports or compatibility problems should be handled via mail as they are rarely fixable via telephone, and should be accompanied by hard copy detailing your system configuration, the software involved, and a the steps required to repeat the problem. Use a copy of the Problem Reporting Log found «behind» this manual for reporting bugs or compatibility problems.

We would also enjoy hearing about any successes you have in using RAMLink with various types of software.

Acknowledgements

RAMLink was originally conceived by CMD in 1987, but the project was postponed because of high RAM prices. Since then, several different versions of RAMLink have been designed and tested by CMD to find the most advantageous design. While cost has always been a factor in the general design, it has never effected decisions concerning the reliability or effectiveness of the final design. Compatibility, as always, has been a strong determining factor in all phases of the design.

Concerning those responsible at CMD for the design of RAMLink, CMD founder Mark (Codehead) Fellows conceived the idea and has spearheaded the project. Mark was responsible for most of the hardware and firmware design, helped at various times by Paul Bosacki. Paul has also been responsible for creating routines for GEOS compatibility both within the DOS, and also

conceived and created the gateWay, our alternative to the GEOS deskTop. Charles R. Christianson (Charlie Sr.) handled mechanical drawings and case design. Doug Cotton was responsible for testing and documentation.

We would like to express our appreciation to Peter Fiset of Performance Peripherals, Inc., who also assisted in parts of the hardware design of RAMLink. Thanks are also due to all who participated in beta and gamma testing of RAMLink. A special note of thanks to Julie, Karen, Ruth Ann, Noelle and Roxane for their support and for allowing us to stretch the 'for better or for worse' part of our marriage vows.

Throughout the project, we have constantly listened to users' ideas and concerns, and wherever possible have implemented designs or design changes to reflect their desires. While this approach has sometimes delayed bringing the product to market, it reflects the design philosophy of CMD: that products should as closely match the needs and expectations of the market which the product is for. Because of this, not only CMD, but also those users who have constantly kept us appraised of their thoughts, are responsible for the final product.

Section 2
Getting Started

Before you Begin

Before you attempt to attach and begin using your RAMLink system, you should read this manual. It may not be necessary for you to read it completely, but you should at least browse through it and become familiar with the sections which pertain to you and your intended use of the device. More importantly, before attempting to attach RAMLink to your system, read this section completely, and pay special attention to any warnings which apply to you and your system.

Also, before attempting to attach and use your new RAMLink, please note the following warnings:

- If you have JiffyDOS or some other modified Kernal ROM installed in your computer, turn off, disable or remove it before attempting to use RAMLink. Failure to do so could cause undesirable operation. Some ROMs may make it impossible to operate your computer with RAMLink attached due to timing problems.

- Never plug RAMLink into your computer or remove RAMLink from your computer while the computer is turned on. This could cause serious electrical damage to either your computer, RAMLink or both.

- Before placing any RAM expander or cartridge into RAMLink's RAM Port, removing a RAM expander or cartridge from that port, attaching or removing an internal RAMCard, or installing or removing SIMMs, turn off your computer and remove all power from RAMLink, including the backup battery if you have one attached. Attaching or removing devices from the RAM Port while it is powered could cause serious electrical damage to RAMLink, RAMCard, SIMMs or your RAM expander.

- Never plug any cartridge into RAMLink's Pass-Thru port while your computer is turned on. This port is powered directly by the computer. Plugging cartridges into this port while the computer is turned on could result in electrical damage to your computer, RAMLink, the cartridge, or any combination of these products.

- Never turn on your computer with RAMLink attached while RAMLink's power supply is turned off or disconnected, especially if you have a backup battery attached to RAMLink. This will discharge the backup battery almost instantly and could cause further damage or a hazard.

- Do not attempt to operate a CMD HD series drive on the parallel port unless you have installed HD DOS version 1.64 or higher. Earlier versions of HD DOS do not support parallel communications.
- Do not attempt to operate any of the CMD HD series drive utility programs which require you to place the drive into CONFIGURATION MODE or INSTALL MODE with RAMLink enabled if you have a parallel cable attached. These modes use the ROM based version of the operating system which does not support parallel communications.

HD Owners

Before proceeding with the setup of RAMLink, install the new version of HD DOS located on the RAMLink UTILITIES disk, which should be located in the front pocket of this manual. Do this by loading and running the program REWRITE DOS. The instructions for this operation can be found in APPENDIX A of this manual. If you are already using HD DOS version 1.64 or higher, this step is not necessary.

You may also update your most recent copy of the HD UTILITIES disk by scratching the SYSTEM HEADER file and the HDOS VX.XX file from that disk, and by copying the SYSTEM HEADER and HDOS VX.XX file from the RAMLink UTILITIES disk to the HD UTILITIES disk.

Hookup

If you have not yet removed RAMLink from its shipping carton, do so now, and remove it from any protective wrapping materials. You should also unpack the power supply and place it in a well ventilated, out-of-the-way location. Be sure to keep the shipping carton in case you have any problems with the system and find it necessary to return the unit to CMD for any reason.

Connections

Please follow the instructions given here when connecting your RAMLink unit for the first time. Do not skip any sections unless they do not pertain to you (don't get ahead of us - it could cause serious problems). Whenever necessary, refer to the figures and diagrams given throughout this section for assistance in locating ports, switches and indicator lamps.

Installing SIMMs

If you have purchased additional SIMMs (memory modules) for your RAMLink unit, you must make sure that your RAMLink is equipped with a RAMCard. If it is, see Appendix F for instructions on installing SIMMs.

Installing RAMCard

If you ordered RAMLink with memory installed then it should already contain a RAMCard, and no further installation is required. You may verify this by looking into the PASS-THRU port. Near the front of the port you should be able to see the edge of a circuit board. This is mounted on a connector and is located approximately one half inch above the main circuit board. If you have purchased a RAMCard separately, see Appendix E for instructions on installing RAMCard.

Attaching RAM Expanders

RAM expansion units are normally attached to RAMLink via the RAM port. This is the cartridge slot located on top of RAMLink nearest the rear of the unit. Currently, this port will support any Commodore REU (1700, 1764 or 1750), Berkeley Softworks GEORAM, or PPI RAMDrive. RAMLink's RAM port can also support expanded versions of the Commodore REUs, as long as the method used for expansion has been done in the most common manner found. Support for expanded GEORAM units has also been implemented, but had not been tested as of this writing.

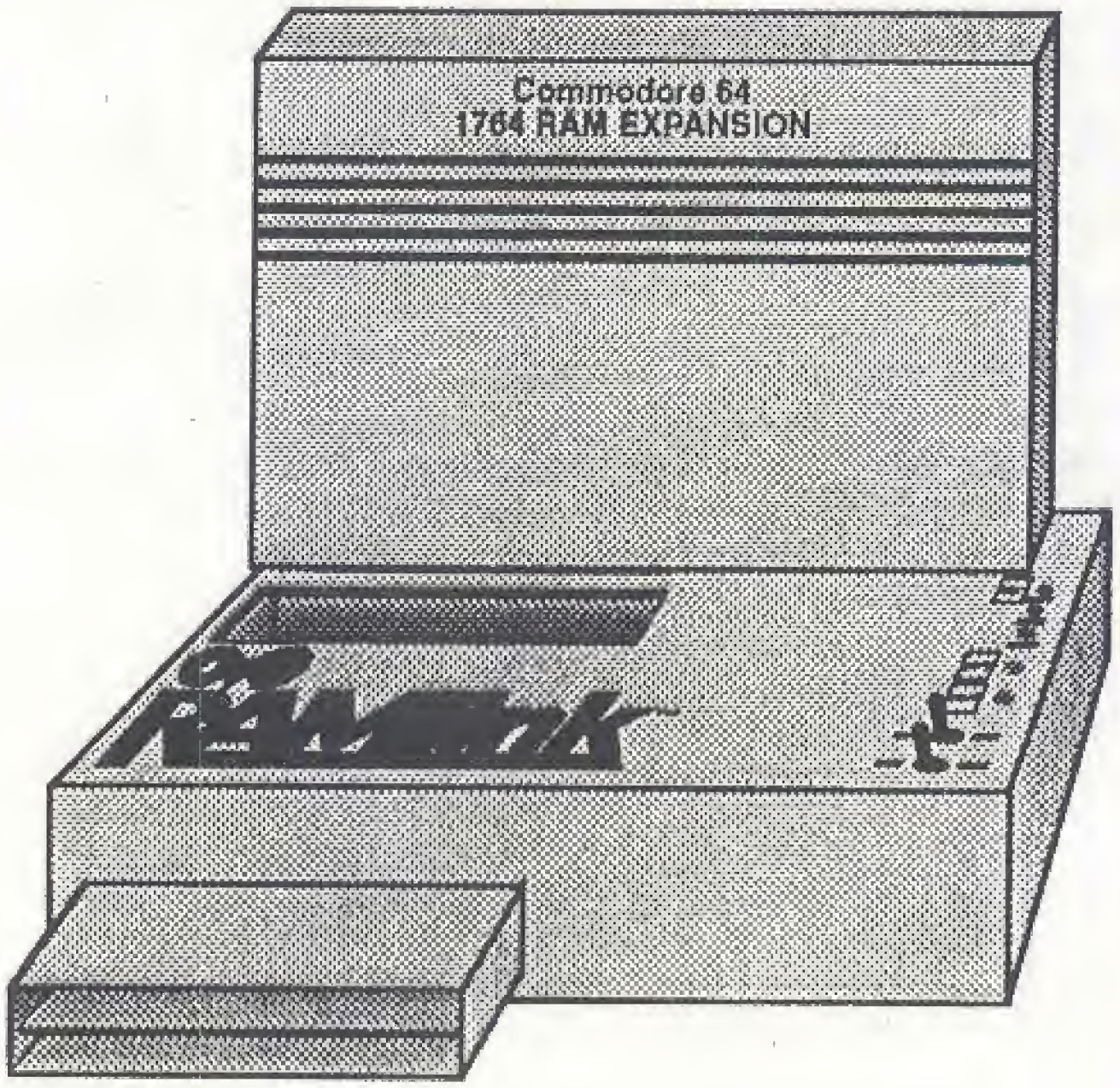


Figure 2-1

If you have a RAM expander which you wish to use with RAMLink, insert it now into the RAM port. When inserting RAM expanders into the RAM port, slide them gently into the port with the label side (top) facing the front of the RAMLink unit. Making sure that cartridge aligns properly, seat it into the connector by pushing down firmly.

WARNING: Never attach anything to the RAM port or remove anything from the RAM port while power is attached to RAMLink. Remove all power first, including the battery.

Attaching RAMLink to your Computer

Before proceeding, make sure that your computer is turned off, and that there are no power connections attached to your RAMLink unit. While it is okay to plug RAMLink into your computer while RAMLink is powered, it is best to avoid doing so when initially connecting RAMLink.

WARNING: You should never, under any circumstances, attempt to connect RAMLink to your computer or disconnect RAMLink from your computer while the computer is turned on. This can seriously damage both RAMLink and your computer.

This is a good time to check your switch positions, before plugging RAMLink into your computer. Make suer that the toggle switch labeled NORMAL / DIRECT is in the NORMAL position. The other toggle switch, labeled ENABLE / DISABLE should be in the position marked ENABLE.

You are now ready to attach RAMLink to your computer. Carefully slide the cartridge port mating connector on the front of RAMLink into the cartridge (expansion) port on your computer. This port is on the far left side of your computer when viewing your computer from the rear. Push firmly to seat RAMLink into the cartridge port connector.

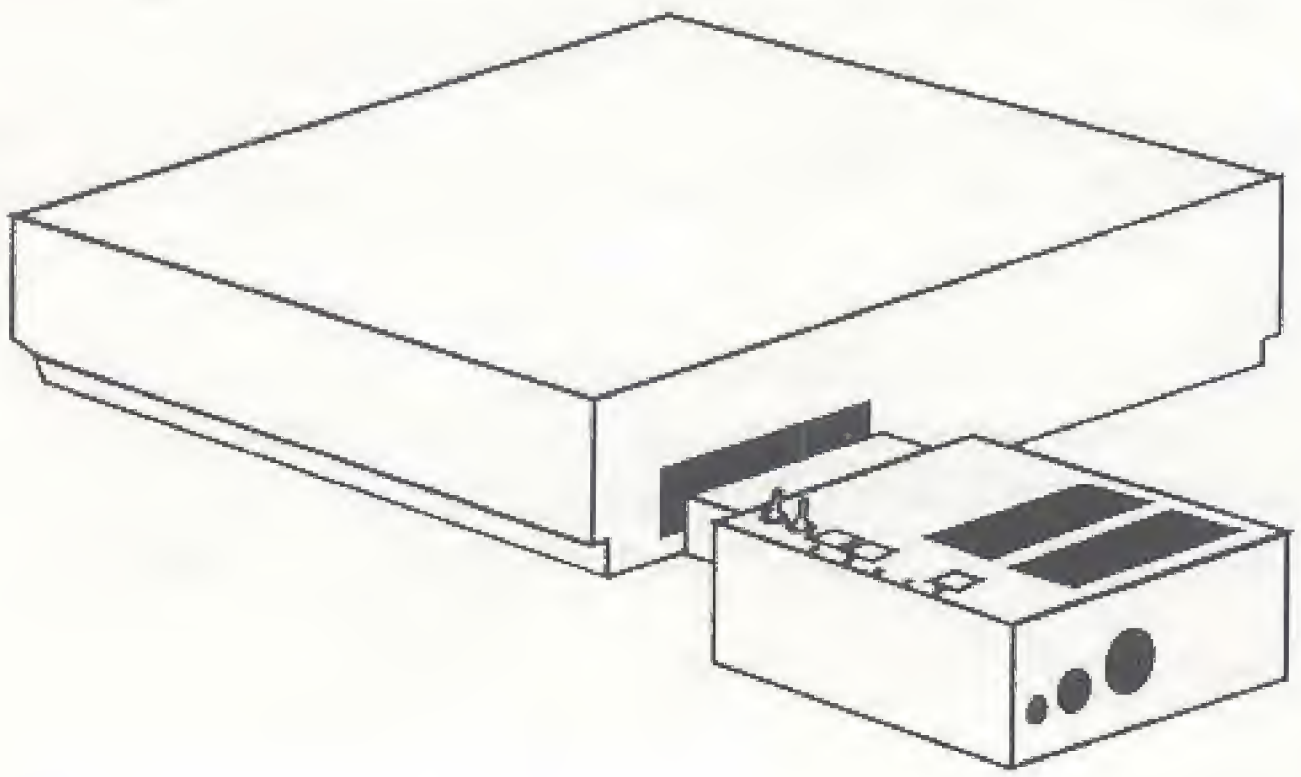


Figure 2-2

Next, plug RAMLink's power supply into an AC outlet. You should use a source which will remain powered at all times if you wish to retain RAM contents whenever your computer is turned off. Make sure that the Mode switches on RAMLink are set to the ENABLE and NORMAL positions, then plug the power supply's connector into the power port located on the rear panel of RAMLink. If you have a backup battery, DO NOT connect it to RAMLink yet. You should only have the power supply and possibly a RAM expander plugged into RAMLink at this time.

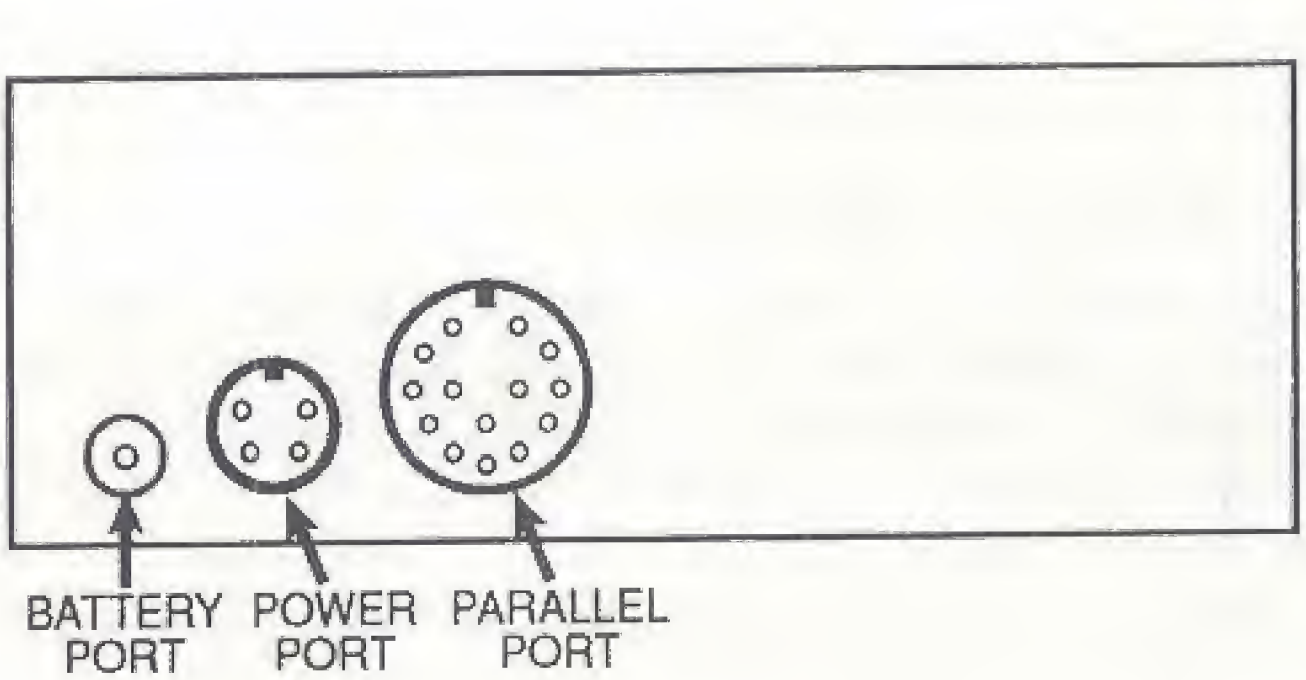


Figure 2-3

You may now turn your computer on (remember that if you have JiffyDOS installed inside your computer that you should have it switched off). There will be a slightly longer delay before you see the initial starting screen whenever you power up your computer with RAMLink plugged in, so don't be alarmed if this takes a second or two longer. This delay will only occur on those occasions when RAMLink is being powered up for the first time after power has been removed from it entirely. If you do not get the initial BASIC startup screen after a few seconds, immediately turn off your computer, unplug the RAMLink power supply from the AC power outlet, and check all other connections before trying to start RAMLink again. If you still do not get a starting screen, go to Troubleshooting Tips at the end of this section for further assistance in tracking down the problem.

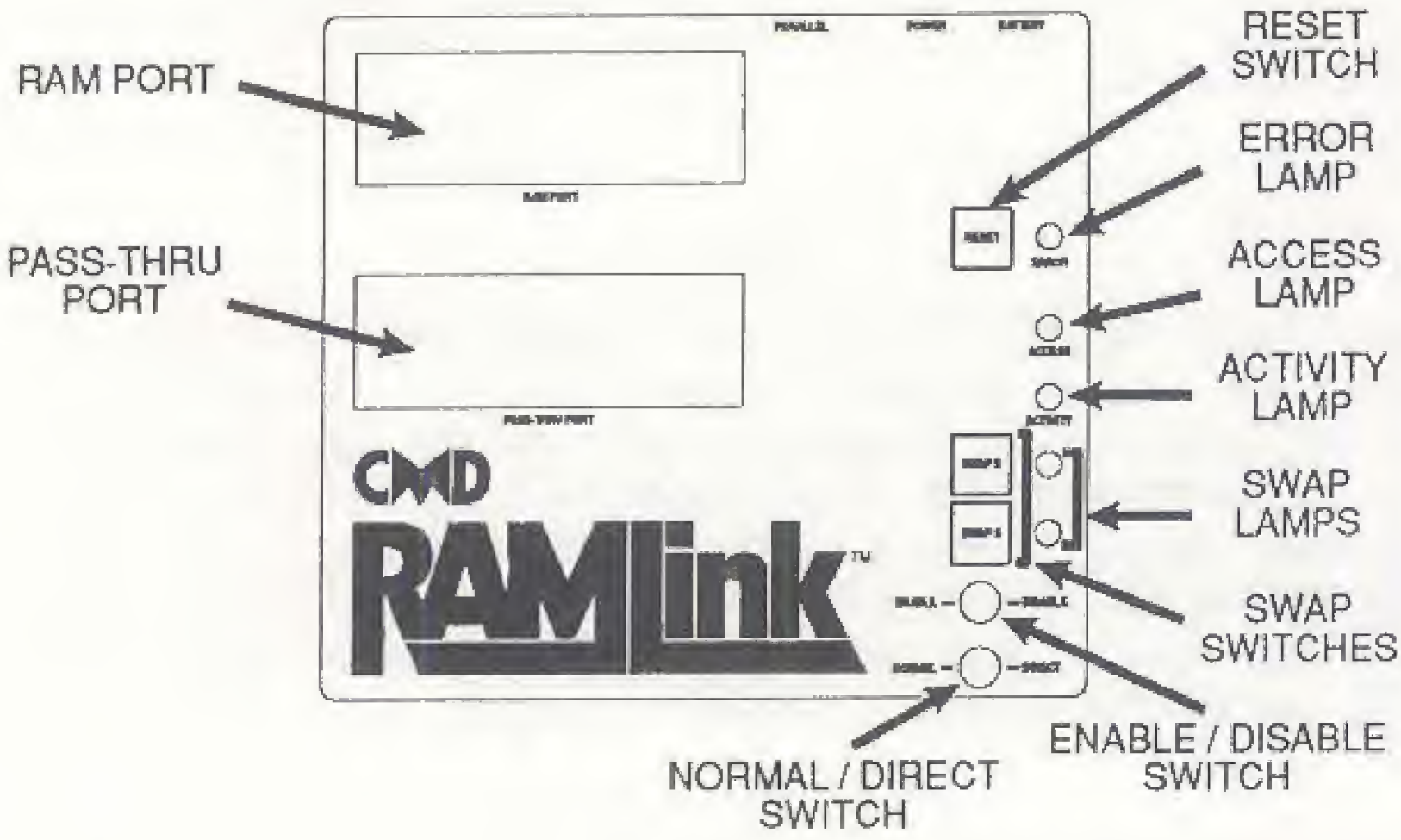


Figure 2-4

Verifying RAMLink Operation

To verify that RAMLink is working, your first indication should be that the standard message which is normally seen on the BASIC startup screen will be slightly different and will contain a JiffyDOS copyright message. If the message which appears does not contain the JiffyDOS copyright message, make sure that RAMLink is enabled (check the ENABLE / DISABLE switch). You may also attempt to read the directory from RAMLink by typing in the following:

```
LOAD"$",16
```

Press RETURN and when you receive the READY. prompt type:

```
LIST
```

Press RETURN and you should see a directory from RAMLink. You can also use the JiffyDOS Control-D function until you see device number 16 appear on the screen, then use the F1 function key to view the directory (see the JiffyDOS instructions for more details on how to do this).

If you have a RAMCard installed in your RAMLink and also have a RAM expander plugged into the RAM port, then RAMLink will have created two separate partitions for these two devices. The first partition will be assigned any RAM in your RAM expansion unit, while the second partition will contain any RAM you have installed in RAMCard. To check the second partition use the procedure given above, substituting "$2:*" for the "$" in the LOAD command.

Backup Battery Hookup

Now that you have verified proper operation, you may attach your backup battery (if you have this option). Make sure that the cable on the battery has been properly attached. The lead with the white line running down the side should be attached to the positive (+) terminal on the battery. After you have verified the cable connections at the battery end, attach the other end to RAMLink by plugging the connector into the battery port on the rear panel.

HD Parallel Hookup

If you own a CMD HD Series hard drive and have the necessary parallel cable, you may attach the hard drive to RAMLink for faster access. Make sure that you have installed HD DOS version 1.64 or higher before attaching the HD to RAMLink's parallel port. First, turn off your computer and all peripherals. You may leave RAMLink plugged in. Attach one end of the parallel cable to the Parallel port on the back panel of RAMLink. The other end plugs into the Parallel port on the back panel of the HD.

To verify parallel operation, remove the serial cable attached to the hard drive, then turn on the HD and your computer. Load a directory from your HD using the device number which is normal assigned to it. After you have verified that the HD/RAMLink parallel connection is operating, turn off your computer and the HD and hook the serial cable back up. By having both the serial and parallel cables attached to the HD, any programs which may need the serial cable to access the drive will still be able to do so. Nearly all software can access the HD via the parallel port as long as drive access is performed with standard kernal ROM routines. Whenever standard kernal routines are used for access, RAMLink will automatically route the access through the parallel port.

Note: Some programs (such as HD-TOOLS) will require you to turn off the parallel access. Using the Parallel Off command (`@P0`) will accomplish this. When you wish to turn parallel back on again, use Parallel On (`@P1`). See the command reference section for more details on these commands.

Partitioning & Configuration

RAMLink perform an auto-configuration when powered up for the first time, or anytime it is powered up after power has been removed. Auto-configuration performs a number of tasks; it checks to see if any RAM is attached, how much, and of what type. It then sets aside a small amount of RAM at the very top of memory, stores a configuration table, creates either one or two partitions (one for each type of RAM attached) and then formats those partitions. RAMLink will allow you to create a number of partitions. After the installation procedures given previously, it is ready to use as is or you may use our utilities to change some of its operating parameters. To take full advantage of RAMLink for your particular needs, it may be necessary for you to delete the partitions created by RAMLink during auto-configuration and create new partitions better suited to your needs. In order to facilitate these changes, the program RAM-TOOLS has been provided on the RAMLink UTILITIES disk. The use of this program is documented in Appendix A of this manual. Please refer to this appendix when using RAM-TOOLS.

Configuration Parameters

The following parameters will be automatically set by auto-configuration, but may be modified by the user via the RAM-TOOLS program:

DEFAULT DEVICE NUMBER	The device number of RAMLink after power on or reset. This will normally be set to 16.
DEFAULT PARTITION	The current active partition after RAMLink is turned on or reset. This is usually set for partition 1.

Partitions

RAMLink may be divided into sections called *partitions*. This is similar to having a number of different disks located within the same physical drive unit. Here are some general guidelines about partitions on RAMLink:

A. There must be at least one partition present on RAMLink in order for it to be usable as a RAM disk.

B. You may create up to 31 partitions.

C. Partition size is dependant upon the partition type. Emulation Mode partitions contain the same number of free blocks as the drive they emulate. Native mode partitions are variable in size and may be from 256 blocks to 65280 blocks in increments of 256 blocks.

Troubleshooting Tips

The following tips should assist you if you have problems with RAMLink. Bear in mind that, as of the time of this writing, RAMLink is still a very new device; therefore, it is difficult to foresee every possible problem.

PROBLEM: BLANK SCREEN - COMPUTER DOES NOT BOOT

POSSIBLE CAUSES AND CORRECTIVE ACTION:

The first thing to do in this instance is to turn everything off and unplug RAMLink's power supply from the AC outlet. This kind of problem is often caused by forgetting to plug something in, or by having something incorrectly attached. Other problems may lead to faulty cartridges, expanders, or even a bad RAMCard or RAMLink ROM. The best general approach is to disconnect anything extra you have attached to RAMLink and try again. If RAMLink works at this point, start adding items back on one by one until you find the source of the problem. You might also see this happen if you power up RAMLink for the first time after power has been removed and have an REU plugged in with the RAMLink ENABLE / DISABLE switch in the DISABLE position. Another possible cause is corrupt RAM contents, either in the computer itself or in RAMLink. Make sure you leave your computer turned off for a few minutes before trying again. If this fails, remove power from RAMLink for an extended period of time (20 or 30 minutes) and try again.

- Computer or power supply is not plugged in or power strip turned off
  Action: Check all connections and switches
- RAMLink is not properly connected (i.e., crooked insertion)
  Action: Check or remove and re-insert
- RAMCard not properly installed or defective
  Action: Remove and re-install or check unit without RAMCard
- Cartridge in Pass-Thru port improperly connected, incompatible or inoperative
  Action: Remove cartridge, check operation, re-install and check again
- RAM expander in RAM Port defective or improperly connected
  Action: Remove expander, check operation, re-install and check again
- Bad ROM in RAMLink
  Action: Check error light (a bad ROM will usually activate it)

PROBLEM: COMPUTER BOOTS - RAMLink DOES NOT OPERATE

POSSIBLE CAUSES AND CORRECTIVE ACTION:

This most often happens when you forget to attach power to RAMLink or when you forget to plug RAMLink's power supply in. You may also have the ENABLE / DISABLE switch in the DISABLE position. Another possibility is a bad ROM.

- RAMLink power supply is not plugged in or power strip is turned off
  Action: Check all power connections and switches
- ENABLE / DISABLE switch is in DISABLE position
  Action: Check switches and correct as needed
- ROM checksum error
  Action: Check error light

PROBLEM: INTERMITTENT OR QUIRKY OPERATION

This covers a lot of possible problems. Bad RAM, timing problems, bad power supply or power source or other hardware problems can all be causes. Software incompatibility can also cause unusual side effects, so be aware of this and test for it if possible by trying other software. Another source of unusual operation could be weak bus signals when using a RAM expander or other cartridges.

The best approach to take in these cases is to try other software, try disconnecting any extra cartridges. You should also run the RAM TEST and HARDWARE TEST programs supplied with RAMLink to determine if problems exist with timing or with any attached RAM. If you have a 1581 or a 1541-II disk drive, you can substitute the power supply from one of these for the one which comes with RAMLink as they are electrically the same.

Section 3
Special Features

Default Device Number

The device number of RAMLink is preset to 16 during auto-configuration. Unlike other peripherals used with Commodore computers, this device number is not controlled by hardware, but is instead kept in a table located in the system partition . You may reassign RAMLink's default device number to any device number from 8 to 29 by using the RAM-TOOLS utility located on the RAMLink UTILITIES disk.

Default Partition

The default partition is the partition which is active after RESET has been pressed or power has been applied and is preset by auto-configuration to partition number 1. You may change the default partition at any time by using the RAM-TOOLS utility. When changing this default, make sure that you choose a partition which is actually in existence on RAMLink.

Autofile Boot Loader

RAMLink contains a special area in the system partition which can be used to define a number of parameters which allow a program to be automatically loaded when your computer is turned on. This file may be located on any drive attached to your system, and may be either a BASIC or a machine language program. Separate areas are maintained for 64 and 128 modes, so 128 users may have separate files for each mode. The utility called AUTOFILE EDITOR, located on the RAMLink UTILITIES disk may be used to create the necessary configuration for programs you wish to load in this manner.

Swapping Device Numbers

One of the most powerful and unique features of RAMLink is its ability to swap device numbers with other disk drives on the serial bus. This function is implemented through the use of the SWAP 8 and SWAP 9 switches on the front panel. Pressing SWAP 8 or SWAP 9 causes RAMLink to change its device number to device 8 or 9 and assigns RAMLink's current device number to the drive that is normally device 8 or 9. The red indicator above the appropriate SWAP button will light when you activate the SWAP 8 or SWAP 9 switch. The actual swap does not take place until the next serial bus access occurs.

As an example, assume RAMLink is device 16 and there is a 1541 drive on the serial bus that is device 8. When SWAP 8 is pressed, RAMLink's SWAP 8 indicator will light up. When the next serial bus access occurs, RAMLink becomes device 8 and the 1541 is swapped to device 16. Pressing SWAP 8 again restores RAMLink and 1541 to their original device number assignments during the next serial bus access.

The most common use for swapping device numbers is when using software that will recognize only devices 8 and/or 9. With such programs, the SWAP function enables you to make RAMLink device 8 or 9 before loading the software (if the program is on RAMLink), or after loading (if the program is copy-protected and cannot be moved onto RAMLink). In either case, having the ability to make RAMLink device 8 or 9 guarantees that any data files used by the program can be stored on RAMLink.

For CMD HD owners, there are a couple of things to remember about device number swapping which are not mentioned in the HD manual. For either of the SWAP switches on the HD to operate properly, you must leave the serial cable attached to the HD. SWAP commands are sent directly over the serial bus from the HD to other devices. Also note that the HD may not initiate a SWAP to RAMLink. If you wish to swap device numbers between RAMLink and a CMD HD, use RAMLink's SWAP switches. Also note that you should never attempt to SWAP the HD to a device number which you are un-swapping RAMLink from. For example, if RAMLink is swapped in as device number 8 and you wish to make the HD device number 8 instead, you must first press SWAP 8 on RAMLink, perform some disk operation, and then you may press SWAP 8 on the HD.

Mode Switches

RAMLink has two mode switches located on the front panel. These switches allow control over RAMLink activity and special features. The following paragraphs will explain how these switches are used.

Enable / Disable Switch

The ENABLE / DISABLE switch is used to determine if RAMLink is active or not. To use RAMLink, this switch must be in the ENABLE position. This switches in RAMLink's kernal ROM, allowing it to effectively replace the kernal ROM mounted in your computer. In the DISABLE position, RAMLink becomes totally inactive, allowing the kernal ROM in your computer to operate as it normally would. The main reason for disabling RAMLink would be to avoid incompatibility with software. While this should be a rare occurrence, it is possible that you may have some software which will not operate with RAMLink attached. If you have a JiffyDOS ROM mounted in your computer, it should be switched off when RAMLink is enabled, but you will want to turn it back on whenever RAMLink is disabled or disconnected.

The position of this switch may be changed while a program is running, but if you are in BASIC, you should turn off the JiffyDOS commands by using the JiffyDOS @Q command before moving the switch to the DISABLE position. If you are switching from DISABLE to ENABLE, then you should also be sure to issue a SYS command to activate JiffyDOS. Please note that the 128 SYS address is different when using RAMLink:

C64 or 64 mode	`SYS 58551`
C128 in 128 mode	`SYS 57651`

Normal / Direct Switch

The NORMAL / DIRECT switch controls how a RAM expander plugged into the RAM Port on RAMLink is connected to the bus. Please note that the NORMAL / DIRECT switch position may be changed at any time as long as there is no RAMLink access or activity going on.

When used in the NORMAL position, the I/O1 and I/O2 lines are disconnected from the RAM Port, thus making it impossible to directly access a RAM expander plugged into that port directly from the computer. This will usually protect you from having the contents of the RAM expander corrupted by software which may attempt to use the RAM expander directly. In this mode, the RAM is controlled only by RAMLink's RL DOS operating system as part of a RAM disk.

When used in the DIRECT position, the I/O1 and I/O2 lines from the computer are connected to the RAM Port. This allows you to use a RAM expander plugged into the RAM Port with software which is able to use that type of RAM expander directly. The RAM expander is still viewed as part of RAMLink when this switch is in the DIRECT position, so it is wise to assign the RAM expander a partition of its own if you have both a RAM expander and RAMCard. This is done automatically by the auto-configuration which occurs when first powering up RAMLink, and this will be identified as partition number 1. Note also that when you place this switch in the DIRECT position, the I/O1 and I/O2 signals will no longer be routed to the Pass-Thru port on RAMLink.

Also note that when using a RAM expander in DIRECT mode, if you have a RAMCard mounted inside RAMLink, the RAM in that RAMCard is still available as a RAM disk. This means that if you are using a program which directly accesses the RAM expander plugged into the RAM Port, you may also use the remaining RAM from RAMCard for normal disk access.

There are four signals which do not connect to the RAM Port from the computer. These are the GAME, EXROM, ROMH and ROML signals. These signals are not used by the RAM expansion units which are accepted by RAMLink in this port.

RAM Port

The RAM port is intended to allow you to connect RAM expansion cartridges from various manufacturers to RAMLink. In so doing, RAMLink can control the memory in the cartridge as if it were part of RAMLink. Power for this port is supplied by RAMLink, and should a power failure occur, the optional backup battery can keep the contents in the RAM expander from being lost for a period of time. To connect a cartridge to this port, turn off your computer, and remove all power to RAMLink (including the backup battery, if you have one). Insert the RAM expansion cartridge with the label facing the front (toward the computer). You may then turn on the computer. RAMLink will automatically configure itself, making partitions for each type of RAM found. Note that only two types of RAM may be attached at any time, and one of these must be a RAMCard unless you have only one type of RAM attached. Approved RAM expanders are the Commodore 17xx series (1700, 1764, and 1750), the Berkeley Softworks GEORAM, and the PPI RAMDrive.

WARNING: Never plug any RAM expansion cartridge into or disconnect any RAM expansion cartridge from this port while your computer is turned on, or while RAMLink is powered (either by its power supply or the backup battery). This could damage the RAM expansion cartridge, RAMLink, your computer, or any combination of these.

Pass-Thru Port

The Pass-Thru port is intended to allow you to connect utility and I/O cartridges to your computer. Power for this port is taken directly from the computer itself, and not from RAMLink. To connect a cartridge to this port, turn off your computer, insert the cartridge with the label facing the front (toward the computer). You may then turn on the computer. The cartridge will now be visible just as if it were directly plugged into the cartridge port on the computer itself.

Not all cartridges are guaranteed to work in RAMLink or with RAMLink enabled. You may have to disable RAMLink in order to use some cartridges. Note that the NORMAL ./ DIRECT switch will have an effect on how this port connects to the computer, specifically, the I/O1 and I/O2 signals are disconnected from this port whenever the DIRECT mode is selected. If you are using a cartridge which has a push-button switch on it to activate it, do not attempt such activation while a RAMLink access is occurring. RAMLink access can be determined by viewing the ACCESS and ACTIVITY indicator lamps on the RAMLink front panel.

WARNING: Never plug any cartridge into or disconnect any cartridge from this port while your computer is turned on. This could damage the cartridge, RAMLink, your computer, or any combination of these.

Reset Switch

This switch can be used to reset your computer, and will at the same time reset RAMLink. 128D owners please note that the Reset switch on RAMLink will not reset the internal 1571 drive located in your computer. Only the computer and drive reset switches located on the side of the 128D can perform a drive hardware reset.

Parallel Port

The Parallel port on the rear of RAMLink is used to connect RAMLink to a similar port on the CMD HD Series hard drives. HD users who desire the fastest performance can easily boost data transfer speed to the maximum attainable level by connecting these two devices together. RAMLink contains special commands to switch the Parallel port on and off in case of incompatibility with certain software. This may happen with software which normally attempts to use burst mode commands, such as CP/M. You should always disable the Parallel port when using software which requires placing the HD in CONFIGURE or INSTALL modes, such as HD-TOOLS, CREATE SYS, and REWRITE DOS. These commands are:

Parallel Off	`@P0`
Parallel On	`@P1`

Battery Connector & Backup Battery

The Battery connector on the rear of RAMLink is used to connect the RAMLink backup battery unit to RAMLink. This battery unit is used to protect the contents of RAM whenever a power loss occurs, or can be used to transport RAMLink from one location to another. Note that it may take several days for the battery to become fully charged, since the charging circuit inside RAMLink charges the battery at a very slow rate. Leave it attached to RAMLink with the power supply attached and turned on for at least a week to assure yourself that a full charge has been attained. The length of time that the battery can protect your RAM from power loss is dependent upon the amount and type of RAM which is mounted in RAMLink. The minimum amount of time should be approximately 8 hours, and the maximum would be in the area of 20 hours. These figures were attained using only stock devices, and does not take into account modified (expanded) REUs or custom-make RAM devices.

Section 4
Partitions and Subdirectories

Partitions

The entire storage area of RAMLink may be divided into a number of smaller areas called *partitions*. While this term is very common to users of larger computer systems, it may be new to many users of Commodore computers. Simply stated, the use of partitions on RAMLink gives the appearance of using a number of separate disks, all located within the same physical device. In all, RAMLink can handle up to 31 of these partitions. As mentioned earlier, RAMLink automatically creates one or two partitions during auto-configuration. Partitions may be created and deleted by using the `RAM-TOOLS` utility. Each partition may be defined to be one of several types. The three main types currently available for use on RAMLink are:

Native Mode
Emulation Mode
Foreign Mode (Direct Access)

In all, RAMLink can handle up to 31 of these partitions. As mentioned earlier, RAMLink creates one or two partitions during auto-configuration. You may change your partitioning by deleting these partitions and creating your own using the supplied RAM-TOOLS utility.

If only one type of RAM is present, only one partition will be created, while two partitions will be created if there are two types of RAM present. The partitions created by auto-configuration will be Native Mode partitions, and if two are created, the first (partition 1) will be made up of the entire amount of RAM located in the RAM expander plugged into the RAM port. The second partition (partition 2) will be made up of most of the RAM contained on the internal RAMCard. If only one partition is created during auto-configuration, this indicates that only one type of RAM has been detected, and this partition will be made up of most of the RAM contents of whichever type of RAM device is being used.

Current Partition

Partition 0 (zero) has a special meaning under RL DOS. It is used to indicate the current partition (the partition in which is currently active). This insures compatibility with software which issues a "0:" within filenames or disk commands. Before attempting to use most commercial software with RAMLink, it is usually wise to select the partition you wish to use. This will assure that any further file access will occur within that partition, especially if the software does not allow you to send disk commands.

Native Mode Partitions

Native Mode partitions allow you to take full advantage of the many additional features provided by CMD's RL DOS (RAMLink Disk Operating System) while retaining full compatibility with all standard Commodore DOS commands. This type of partition can access up to 16 Megabytes of storage space, providing the highest capacity available to Commodore DOS commands on RAMLink. Since Native Mode partition size is variable and may also be as small as 256 blocks, it allows you to use only the amount of storage which you feel is required for a particular partition. This mode is also the only mode which supports true subdirectories and dynamic allocation of directory space. This means you can easily organize your files and continue to add files until no free blocks remain in the partition.

Native mode partitions have 256 sectors per track, and may have from 1 to 255 total tracks. Since the size is variable, all header, BAM, and directory information must be stored on track 1. Subdirectories have a file type of DIR and have been assigned a filetype value of 6.

Emulation Mode Partitions

Emulation Mode partitions allow you to retain compatibility with software programs which require the tracks and sectors of a disk, as well as the BAM and directory, to be laid out in the same way as on a particular type of Commodore floppy disk drive. For this reason, an Emulation Mode partitions has a fixed storage capacity equal to the capacity of the disk drive that it is emulating. There are three types of Emulation Mode partitions available on RAMLink. They are:

1541 Emulation Mode
1571 Emulation Mode
1581 Emulation Mode

As the names imply, these partitions emulate Commodore's popular 1541, 1571 and 1581 disk drives. This is accomplished by utilizing the same track and sector layout as the type of disk drive being emulated. The BAM and directory areas of these partitions are also located in the same blocks as they would be on the emulated drive. Even the internal job queue codes and locations have been duplicated. Other similarities (and a few beneficial differences) have been created within the emulation mode partitions. It is important to note that emulating these other disk drives fully in the areas of hardware and firmware mapping would have driven the cost of RAMLink to an unreasonable level, and was therefore not attempted. All other aspects of compatibility were carefully scrutinized, and incorporated where feasible. The following paragraphs describe each of the individual emulation modes.

1541 Emulation Mode Partitions

In 1541 Emulation Mode partitions the directory and BAM are found in the same locations as they are on the Commodore 1541. All bytes within these blocks have been defined identically to their counterparts on the 1541, including the BAM bytes. This type of partition uses 683 blocks of RAM, of which 664 are free for user data or programs.

1571 Emulation Mode Partitions

1571 Emulation Mode partitions are identical to 1541 Emulation Mode partitions with only a few differences. First, 1571 partitions have twice as much storage capacity as do the 1541 partitions. This type of partition uses 1366 blocks of RAM space, of which 1328 are free for user data or programs.

There are also a number of extra bytes required for the BAM in the header block and on track 53. These have been allocated in the same fashion as they are on the standard 1571 disk drive. Also, 1571 partitions are always equivalent to the double-sided version of the 1571. If for some reason you need to emulate a single-sided 1571 disk, the 1541 partition is fully capable of doing this and should be used instead.

As with 1541 Emulation Mode partitions, it is possible to read and write to the 'super' side sector form of REL files. This method, if used, allows REL files created within 1571 Emulation Mode partitions to grow beyond the former maximum size of 726 blocks. Creating a super side sector REL file can only be done by writing a utility which creates the super side sector itself, then places pointers to any existing side sectors for that file into the super super side sector. The directory entry for the REL file would also have to be modified to point to the super side sector.

Burst commands which are normally accepted by the 1571 are not accepted by RAMLink, and will return a syntax error. This is because the normal burst protocol requires that data be transferred directly via the serial port, which is not physically connected to RAMLink.

1581 Emulation Mode Partitions

1581 Emulation Mode partitions provide emulation of the 1581 header, BAM, and directory information of the 1581. One area of difference between the 1581 and its RAMLink counterpart is that RAMLink does not utilize the track cache buffer found on the standard 1581. This was incorporated into the 1581 to increase speed - an area where RAMLink does not suffer. Burst commands for the 1581 are not supported for the same reason given for 1571 Emulation Mode partitions. All other standard DOS commands are fully implemented, including 1581 style partitioning commands.

In order to retain full compatibility with the 1581, the DOS initialize command initiates a change in the current 1581 partition status, causing further accesses to be performed in the root directory of the 1581 partition.

Foreign Mode (Direct Access) Partitions

Foreign Mode partitions were created by CMD for use on the CMD HD for data from another type of computer. Foreign Mode partitions were included in RL DOS for an entirely different purpose - to protect areas of RAM from use by RL DOS. This type of partition is ideal for those who wish to use the direct RAM access commands provided with RAMLink, or if you have a Commodore or Berkeley RAM expander which you wish to access in DIRECT mode. When you create a Foreign Mode partition on RAMLink, you will not be able to get a directory of that partition, nor will you be able to use that partition with any other DOS commands. Only by using the Direct RAM Access routines discussed later in this manual will you be able to read from and write to this type of partition.

Native Mode Subdirectories

The following information is intended as an introduction to how Native Mode subdirectories are stored on RAMLink. The commands used to create, remove, and move around within subdirectories can be found in the Command Reference section of this manual.

Native Mode subdirectories are similar in structure to the subdirectories used on MS-DOS types of computers. When a subdirectory is created, a DIR type file (filetype 6) is created and added to the current directory. Subdirectory names may be up to 16 characters long, just as any other filename. This "file" is initially two blocks long, and consists of a directory header block and the first directory block. These blocks are always located next to each other on the same track, and if two adjacent blocks cannot be found, no directory will be created.

The storage space available to subdirectories is the same as that available to the parent directory (the directory in which the subdirectory exists). In fact, all of the blocks within a Native Mode partition are shared between all directories within that partition. This is quite different than the method used for 1581 type subdirectories (or sub-partitions as they are referred to in this manual). This means that if there are 62000 blocks free in the partition, this number of blocks free will be indicated no matter which directory you are located in. If a 37 block file is saved in any directory within the partition, all directories within that partition will indicate 37 fewer blocks.

Subdirectories may be created in the 'root' directory (the first or main directory in that partition) or within another subdirectory. Placing a subdirectory within another subdirectory is called 'nesting'. There is no actual limit to the number of directories located in a partition, nor is there any limit to how deep subdirectories may be nested. The only limitation on creating subdirectories is the number of adjacent blocks located within the partition. There are, however, some practical limitations if you wish to be able to easily access files located in various subdirectories with a single

command. This is because the input buffer of RAMLink is only 127 characters long, therefore nesting subdirectories too deep could necessitate using more than one command string to access files within a particular directory.

Also note that when you are in a subdirectory, it is not possible to use the DOS NEW command on the partition containing that subdirectory. This was done to protect against accidental erasure due to user error. Other protective limitations have been place on subdirectories - you cannot delete a subdirectory while there are still files located within that subdirectory, and the command for removing a subdirectory may only be issued from the parent directory of the subdirectory you wish to delete. These concepts are explained further in the Command Reference Section of this manual under the individual subdirectory commands.

Section 5
JiffyDOS

About JiffyDOS

JiffyDOS is a combined disk drive speed and DOS command enhancement system. Normally JiffyDOS is sold as a two-part system, replacing the kernal ROM(s) in the computer and the DOS ROM in a disk drive attached to your computer. By replacing both components which control the DOS transfers within the system, it is possible to speed up all file access with a high degree of compatibility.

RAMLink contains the computer portion of the JiffyDOS system. By integrating the JiffyDOS kernal routines into RAMLink in this manner, it becomes possible to use the JiffyDOS DOS wedge and command enhancements with all drives on the system. However, unless you add JiffyDOS drive ROMs to the disk drives on your system, there will be no real speed enhancement when using your drives.

A JiffyDOS manual has been included with your RAMLink manual so that you may become familiar with the JiffyDOS commands and enhancements. Since this manual is normally applies to the regular JiffyDOS system, there are a few differences with the RAMLink version which are documented here.

- References to the JiffyDOS switch may be assumed to be equivalent to the RAMLink ENABLE / DISABLE switch
- The SYS commands given in the JiffyDOS manual which are used for switching to the JiffyDOS kernal while in BASIC and for re-enabling the JiffyDOS function keys are the same for the C64 (or a C128 in 64 mode), but are different for a C128 in 128 mode. These SYS commands for RAMLink are:

C64 or C128 in 64 mode	SYS 58551
C128 in 128 mode	SYS 57651

IMPORTANT NOTE: If you have JiffyDOS on your system already, you must switch it off on your computer when using RAMLink with the ENABLE / DISABLE switch in the ENABLE position. If you disable RAMLink for any reason, you may re-enable the JiffyDOS in your computer using the directions and SYS commands given in the regular JiffyDOS manual.

Section 6
Using Software

General Software Installation

There are many types of programs, but for this discussion we will separate programs into two main categories - those which are copy-protected, and those which are not. Most commercial programs, with the exception of games, are compatible with RAMLink in one form or another. This section of the manual will help you to discover which programs will work with RAMLink and the steps required to get them working.

Software without copy-protection

Software which is not copy-protected can almost always be installed directly on RAMLink. With these programs the main concern is usually which type of partition to use.

The best way to determine which partition type to use is to experiment, starting with a Native Mode partition. It is usually best to start by creating a Native Mode subdirectory to hold the files used by the program. Use RAMLink's MD (Make Directory) command to do this and then copy all the files from the program disk to that subdirectory by using FCOPY.

To test whether the program works or not, change to the partition and directory where you have placed the program and begin running it as you would from a floppy disk. It may be necessary to set RAMLink as device 8 or 9 by using the SWAP switches on the front panel (most software requires device number 8). If the software does not work from within a subdirectory, try using the root (or main) directory in a Native Mode partition.

Some software, even though it contains no copy-protection, will only work with a certain type of drive. Such programs will usually work in one of the Emulation Mode partitions on RAMLink. Most often, the partition will be a 1541 or a 1571 Emulation Mode partition, though many programs may work in, or possibly require, a 1581 Emulation Mode partition.

When trying emulation mode partitions, work your way 'downward', trying a 1581 partition first, then a 1571, and finally a 1541 partition. Before testing, be sure to use the CP (Change Partition) command to select the partition in which you placed the program. If the software will still not work in any of the partitions, all is not lost, as there may be some 'hidden' data on the program disk which cannot be duplicated with FCOPY. In this case, use MCOPY to copy the entire disk to a partition of the same type.

If the program still does not work, the software is probably performing some very drive specific tasks. This is rare unless the program is some kind of disk utility. The program might also be making use of a fast loader routine written specifically for a certain type of disk drive. Fast loaders should be disabled if possible in order to get software to load directly from a device like RAMLink.

Multiple Disk Programs

Some programs are distributed on a number of disks. In many cases, simply copying all the files from each of the disks into a single partition using FCOPY will allow you to use this type of program on RAMLink. Watch out for file names which are the same when copying these types of disks. These files may or may not be identical. As long as a multiple disk program uses standard files, and as long as it can determine that the proper disk is in use while searching for a file, this method will work.

If some of the information for a program is stored directly on the disk without a file name, or the disk name is checked to determine if the correct disk is being used, it will be necessary to use MCOPY to copy the disks to partitions. Also, you may only be able to use one of the disks in the set on RAMLink if there is no provision for sending disk commands before a disk swap is to occur. The CP (Change Partition) command is the only way to move from one partition to another on RAMLink.

Other Solutions

Some software expects to find the disk directory in a certain place on the disk. Normally this kind of software can be operated using an emulation mode partition of the required type. It may also be possible for this software to be operated in a Native Mode subdirectory which has been specially created to simulate the directory of a 1541 or 1581 disk drive. These types of subdirectories can be created in an empty Native Mode partition which has the required number of tracks by using the `1541SUB` and `1581SUB` utilities supplied with RAMLink.

Creating an emulation subdirectory is often a good method to use to extend the amount of usable disk space with programs which must find the directory in the same area of a disk as it would on a 1541/1571 or 1581 disk drive. Be aware, however, that some programs check to see how many tracks or sectors it can access, and determine the type of drive being used by doing so. As a result, it may be difficult to determine which subdirectory type to emulate (1541/1571 or 1581). As long as you use Native Mode partitions with 40 or more tracks, 1581 emulation subdirectories will usually work with these types of programs. Some programs will always assume a 1541 or 1571 no matter how many tracks you have in the current partition.

Whenever you use a subdirectory for emulation purposes, remember to use the CD (Change Directory) command to make that directory the current one before using your software. This may be done from another partition, as long as you include the proper partition number and path in the command.

Copy-Protected Software

Normally, copy-protected of software cannot be placed directly onto RAMLink. You may, however, be able copy some of the more mildly protected programs by using MCOPY. This tends to work more often with software supplied on 1581 disks than with 1541/1571 programs.

If a program will not load from RAMLink, you may be able file copy most of the data files to RAMLink with FCOPY, and then load the program initially from the floppy disk. After the program stops loading (and is past checking the copy-protection) you could press the SWAP 8 switch to substitute RAMLink in place of the floppy device. Any subsequent disk access will be directed to RAMLink.

Some programs do not access the disk once they have loaded. If a program is copy-protected and has no need of a disk drive after it has been loaded, there is not much chance of using it with RAMLink. The only method left to try with this type of software is use a memory capture type of cartridge to save the program as an unprotected file. There are also some copy programs (such as Maverick) which will remove protection from programs by using a parameter disk. This is also a good method for making RAMLink bootable copies of protected software. Be aware that not all parameters will remove copy protection; many make exact copies with the protection still intact.

Some Specific Applications

While it is impossible to provide specific details on how to use a lot of particular software titles on RAMLink, there are a couple of alternative operating systems for the C64 and/or C128 which deserve special attention. The majority of applications will require that you experiment with RAMLink to discover what can and cannot be used.

GEOS

GEOS can be used with RAMLink, but because GEOS is a very device specific program, special patches are needed to allow GEOS to recognize RAMLink. These patches have been created by CMD in the form of gateWay, a replacement for the GEOS deskTop. This will allow you to use RAMLink not only as a normal RAM disk under GEOS, but also allows for the use of some of the standard partition types provided by RAMLink. Many other benefits are realized by using gateWay, features which go far beyond a minimal level of support for RAMLink. For more details, see the gateWay manual included with gateWay.

CP/M

CP/M may be used to a limited degree with RAMLink. CP/M contains special drivers for each device it supports. Since most of those devices are accessed by using burst mode commands, and RAMLink is not connected to the serial bus for this type of emulation, only drivers which have been written to access drives via direct kernal or DOS calls can be used. In our testing, only one driver was found capable of working properly with RAMLink. This is the 1541 driver, and when using FORMAT.COM from CP/M, you should format a 1541 partition on RAMLink using the 128 Single Sided format. HD owners should always issue the Parallel Off (`@P0`) command before booting CP/M if a parallel cable is attached to the HD.

JiffyMON

CMD's JiffyMON V2 machine language monitor program can be used with RAMLink. Please note that a different *SYS* command is needed to start JiffyMON when RAMLink is enabled:

```
SYS 58564
```

Section 7
Command Reference

Command Syntax

This section documents many of the commands which can be used with RAMLink. The syntax for these commands is given in a standard format which should allow you to easily recognize required and optional parameters. Examples are used throughout to assist you in determining proper usage of the commands. If any problem should arise in determining command syntax, be sure to check the following information.

Command String Elements

The command string is made up of a number of elements. In the case of commands sent directly from BASIC, the first part of the command string is usually the command itself. In the case of commands sent via the command channel, the command itself is usually found at the beginning of a string sent to RAMLink. The elements of the command string as used in this manual are described below:

Literals are characters which must be entered exactly as shown. These will appear as plain text.

User supplied values are those which must be supplied by the user and whose values and type are dependant upon the use of the command. These will appear as *italicized* text.

Optional parameters and **options** are values or literals which need not be included in the command unless the user wishes to specify the option. Often, the optional parameters will be substituted with a certain default when left out of the command string. Optional parameters will appear within [brackets].

Choice parameters allow or require you to choose from more than one parameter to be placed within the command string. Whenever these appear in the syntax of the statement, the choices will be enclosed by {braces} and the individual choices will be separated by a vertical bar character (|). Only one choice parameter may be used in command string.

You may occasionally notice syntax in which one or more elements is followed by three periods (...). This means that the parameter last shown may be repeated. This will normally be discussed in more detail in the text describing use of the command.

Example command string

The following example illustrates the command syntax used throughout this manual:

```
HEADER"partname"[,Iid][,Dn][{ON|,}Udv]
```

In this command string, HEADER is the actual command and is entered literally. The commas (,) and the 'I', 'D', 'ON', and 'U' are literals contained within brackets and are optional. The parameters 'partname', id, n, and dv are user supplied variables. These variables are described in a table following the command syntax. The table for this command string would look like this:

where: partname = the name you wish to appear in the partition header
id = a two character ID for the partition header
n = the partition number you wish to format (0 or 1)
dv = the device number of RAMLink

Since these parameters are all enclosed in brackets, they are optional. If you decide to use one or more of the optional parameters, notice that they are accompanied by a literal. This literal must be included in the order shown along with the user-supplied variable. Also notice that the 'ON' and a comma are enclosed in braces and are separated by a vertical bar. This means that if you want to supply the unit number, it must be preceded by either the literal 'ON' or by a comma. The full use of this command is shown in the following command string:

```
HEADER"PARTITION 1",IP1,D0,U16
```

Since partition (or drive) zero (0:) is assumed in BASIC 7.0 commands, It would also be possible to shorten this command to:

```
HEADER"PARTITION 1",IP1,U16
```

Paths in Command Strings

Throughout this manual you will see commands which have a [path] shown in the command syntax. Paths are only used when accessing Native Mode partitions, and specify which subdirectory the disk operation is intended for. Here is an example of a command with a path in its syntax:

```
VERIFY"[[n][path]:]filename",dv[,sa]
```

Most of the common commands used on RAMLink will allow you to include a subdirectory path within the command string, provided that the target file is within a Native Mode partition. This path immediately follows the partition number within the command string and is shown throughout this manual within command syntax descriptions as [*path*]. When including one or more subdirectories within a command string, each subdirectory placed in the command must be bracketed between slash (/) characters, and the final slash must usually be followed by a colon.

For example, if you had a file named COPY located in a subdirectory named UTILITIES in partition number 1, you could load this file with the following command:

```
LOAD"1/UTILITIES/:COPY",16
```

The portion of this command which makes up the path is:

```
/UTILITIES/
```

If you had nested subdirectories, and the file COPY was located in a subdirectory named COPIERS which in turn was located within as subdirectory named UTILITIES in partition 1, you might load this file with:

```
LOAD"1/UTILITIES/COPIERS/:COPY",16
```

The portion of this command which makes up the path is:

```
/UTILITIES/COPIERS/
```

You may be able to shorten this command, depending on which directory and partition you are currently located in. For example, if you are already located within partition 1, and the root directory is your current directory, you could skip the partition number in the command string:

```
LOAD"/UTILITIES/COPIERS/:COPY",16
```

If your current partition is partition 1 and your current directory is UTILITIES, you could simply enter:

```
LOAD"/COPIERS/:COPY",16
```

If your current partition is partition 2 but your current directory in partition 1 is UTILITIES, you could enter:

```
LOAD"1/COPIERS/:COPY",16
```

If your current partition is partition 2 but the current directory in partition 1 is COPIERS, you could enter:

```
LOAD"1:COPY",16
```

There is another syntax which will always allow you to begin your path at the root directory. This is helpful when you are located within a different subdirectory in the same partition. For example, if your current directory is GAMES in partition 1, and you wish to load the COPY program shown in the previous example, you can begin your path with two slashes:

```
LOAD"//UTILITIES/COPIERS/:COPY",16
```

Two slashes placed at the beginning of a subdirectory path indicates that the path must begin at the root directory. If you are in a different partition, or

are not sure which directory is the current directory in the partition you wish to access, it is usually wise to use the double-slash method. Remember to include the partition number if you are in a different partition:

```
LOAD"1//UTILITIES/COPIERS/:COPY",16
```

Subdirectory Paths Using JiffyDOS Commands

The examples given above are shown using the standard Commodore DOS methods of loading files. Since RAMLink is equipped with JiffyDOS, you may use the JiffyDOS load commands. Here are the above examples converted to their JiffyDOS equivalents:

```
/"1/UTILITIES/:COPY",16
/"1/UTILITIES/COPIERS/:COPY",16
/"/UTILITIES/COPIERS/:COPY",16
/"/COPIERS/:COPY",16
///UTILITIES/COPIERS/:COPY
/1//UTILITIES/COPIERS/:COPY
```

You may also leave out the quotes (") and the comma and device number by using the JiffyDOS <CONTROL><D> function to change the default device. The last two examples illustrate this option and show the only situation under which you will use three slash characters in a row. The first slash character is assumed to be the BASIC LOAD command by JiffyDOS. You may also use any of the other JiffyDOS wedge commands for loading and saving programs in conjunction with subdirectory paths. See your JiffyDOS manual for more information about these commands.

Sending Commands from BASIC

Most of the commands you send to RAMLink will be from BASIC. This requires very little new knowledge since RAMLink accepts standard disk drive commands. You should note that in order to maximize your use of RAMLink, it may be desirable to use BASIC 2.0 or DOS command channel commands when operating RAMLink on a C128 instead of using BASIC 7.0 commands. This is because BASIC 7.0 places some limitations on the use of partition numbers in its command syntax. Specifically, the drive number parameter in BASIC 7.0 commands may only be represented by a zero (0) or a one (1). Since RAMLink uses this number as an indication of which partition is to be used for the particular command, and since RAMLink can have up to 31 partitions, the BASIC 7.0 commands may or may not be able to access the desired partition. Also, because of the way in which BASIC 7.0 sends commands, it is not possible to include subdirectory paths within these commands. If you wish to include subdirectory paths, use the BASIC 2.0 or DOS command channel version of the command instead.

The Command Channel

Many of the commands discussed in this section, require you to send the command via RAMLink's command channel. Opening a command channel to RAMLink requires the following BASIC statement:

```
OPENlf,dv,sa
```

where: lf = the logical file number
dv = the device number of RAMLink
sa = the secondary address

The logical file number can be any number from 1 to 127. Other numbers are legal (128-255) but will cause side effects which are not usually desirable, so it is wise to avoid them.

The device number is the same as the device number currently assigned to RAMLink. This is set to 16 at the factory, but may be changed to any number from 8 to 29 by using RAM-TOOLS, or set to 8 or 9 by using the SWAP feature.

The secondary address is often referred to as a channel. Secondary addresses 0 through 14 are used to open files, whereas secondary address 15 tells RAMLink that data sent via this channel should be interpreted as commands and command data. Thus, channel 15 is referred to as the 'command channel'. This example shows how to open the command channel :

```
OPEN15,16,15
```

The command channel may be opened from within a program or in 'direct' mode. Whenever you enter a command to be executed immediately, without preceding it with a line number, it is considered to be entered in BASIC's direct mode. Here is an example of sending a command via the command channel:

```
OPEN15,16,15:PRINT#15,"I":CLOSE15
```

This type of command may also be sent without using the PRINT# command as shown below:

```
OPEN15,16,15,"I":CLOSE15
```

Some commands require that other parameters be sent to RAMLink in the form of character strings with the CHR$ function. These types of commands can be sent using the PRINT# command. An example of this is the device number change command:

```
OPEN15,16,15:PRINT#15,"U0>";CHR$(10):CLOSE15
```

The semi-colon which appears between the portion of the command in quotes and the character string code is optional. Some commands require that numeric variables or actual numbers be used as command parameters. This is more common with direct access commands. An example is given in the following U1 command:

```
OPEN15,16,15:PRINT#15,"U1";2;0;1;34:CLOSE15
```

The numbers shown following the semi-colons in the command given above could also be expressed as variables. For example:

```
OPEN15,16,15:PRINT#15,"U1";C;D;T;S:CLOSE15
```

If numbers are used in a command, they may be included within the string portion as long as each is separated from the string and the other numbers by a space:

```
OPEN15,16,15:PRINT#15,"U1 2 0 1 34":CLOSE15
```

In this case, the trailing quote is placed after the last number or parameter required by the command. You may optionally place a colon at the end of the command itself, no matter which way the command is used:

```
OPEN15,16,15:PRINT#15,"U1:";2;0;1;34:CLOSE15
OPEN15,16,15:PRINT#15,"U1: 2 0 1 34":CLOSE15
```

Reading Disk Errors

Disk errors are most often detected by reading the command channel. This is done by using either the GET# or INPUT# commands. Using INPUT# is the fastest method of returning error information from RAMLink. GET# is more commonly used for getting non-error information. Since both the GET# and INPUT# commands must use the BASIC input buffer, they cannot be used in direct mode.

Usually, programs will check for errors immediately after attempting to perform a disk access or disk command. Here is a short program which shows how the error channel is read:

```
10 OPEN15,16,15:INPUT#15,E,E$,T,S:CLOSE15
20 PRINTE,E$,T,S
```

As you can see, four parameters are returned via the command channel when checking for an error. These are, in order: the error number, the error message, the track where the error occurred, and the sector where the error occurred. Many particular errors will not occur at any particular track and sector, in which case the track and sector variables will contain zeroes.

There are occasions where it is more desirable to obtain error data or other information from the command channel one byte at a time. In these instances a program similar to the one that follows could be employed:

```
10 OPEN15,16,15
20 GET#15,E$:PRINTE$;:IFST<>64THEN20
30 CLOSE15
```

If you are using JiffyDOS, it is possible to read the error channel without using a program. This is done by pressing the commercial at (@) key and then <RETURN>.

In the preceding example, we used the status variable to determine when the end of the file was reached. The BASIC status variable ST is quite useful in serial device access, so here is a breakdown of the individual bit values. These definitions apply when the specified bit is set, or equal to one.

BIT	DESCRIPTION OF STATUS	STATUS VALUE
7	DEVICE NOT PRESENT or END OF TAPE	128
6	END OF FILE (EOI)	64
5	CASSETTE CHECKSUM ERROR	32
4	VERIFY ERROR or CASSETTE READ ERROR	16
3	DATA BLOCK TOO LONG (Cassette)	8
2	DATA BLOCK TOO SHORT (Cassette)	4
1	TIME OUT ON LISTENER	2
0	TIME OUT ON TALKER	1

Figure 10-1

The C128 provides another method of checking for disk errors via two reserved BASIC 7.0 variables: DS and DS$. The variable DS returns the error number, while DS$ returns the error message string. These variables can be viewed with a PRINT statement, as shown in the following example:

```
PRINTDS,DS$
```

It is not necessary to print both of these variables at the same time, but the error message string will usually provide help in understanding why the error occurred, and also makes it easier to look up in the error descriptions located in Appendix B of this manual.

Note: DS and DS$ are usually valid only after a BASIC 7.0 disk command.

Partition Numbers in File Names

A partition number may be specified within a filename in place of the drive number. The drive number is the number which can precede the colon (:) in Commodore DOS filenames. Often, this number is not included since it is normally used only with dual drives to determine if the file operation should be directed to drive 0 or drive 1. Whenever this drive number is left out of the command, drive 0 is assumed. These rules also apply to RAMLink, which contains a single RAM disk with up to 31 partitions. Whenever you wish to perform a command or file operation with the current partition, you may use a 0 or leave out the partition number entirely. However, if your current partition is different than the partition in which you wish to perform the file operation, you must either move to that partition first (with the 'CP' command), or specify the partition number within the filename. The following examples should help to illustrate this:

```
LOAD "1:MCOPY",16
OPEN2,16,2,"3:TESTFILE,S,W"
```

Load commands may be abbreviated if you are using JiffyDOS:

```
/1:MCOPY
```

Partition Numbers in Disk Commands

As described above, partition numbers may be used to replace the drive number in filenames. Partition numbers may also be used in this manner when sending disk commands. In fact, any disk command which allows inclusion of a drive number (with the exception of direct access commands), will also allow you to substitute a partition number in its place. You may use partition numbers when formatting, copying, renaming, scratching, validating, and initializing. This is a large part of what makes RAMLink so compatible with Commodore DOS.

In the case of direct access commands, the current partition is assigned to the direct access file when when that file is opened. As a result, you must use the 'Change Partition' command to select the desired partition before opening a direct access file. Once the file has been opened, all commands sent to that direct access file will refer to the partition you were in when the file was opened, even if the current partition is changed.

Partition Commands

Many of the commands used on RAMLink are partition related and, among other things, are used to format, initialize and change partitions.

Creating Partitions

Partitions are created by using the RAM-TOOLS program supplied with RAMLink. Use of this program is documented in Appendix A.

Creating 1581 Style Sub-partitions

You can create 1581-style partitions within 1581 Emulation Mode partitions on RAMLink. This type of partition is sometimes referred to as a subdirectory by 1581 users. Because of the way these partitions allocate space, we feel the term subdirectory does not apply. Since RAMLink is already divided into partitions, and since these 1581 'subdirectories' are nested into 1581 Emulation Mode partitions, we have opted to call them sub-partitions.

Certain limitations apply when creating sub-partitions that are intended to store files. Because of the way physical tracks are handled in Commodore's 1581 DOS, and because sub-partitions must contain header and directory blocks, the minimum size of a sub-partition is 120 blocks. The starting sector must be zero, and the ending sector must be a multiple of 40. Sub-partitions are not allowed to begin on, end on, or contain within themselves track 40. Here is the syntax required to create 1581 style sub-partitions:

```
OPENlf,dv,15:PRINT#lf,"/[n]:partname,"CHR$(st)
CHR$(ss)CHR$(sl)CHR$(sh)",C":CLOSE15
```

where:

lf	=	the logical file number for the command channel
dv	=	the current device number assigned to RAMLink
n	=	the target 1581 Emulation Mode partition
partname	=	the name of the 1581 sub-partition name to be created
st	=	the starting track of the sub-partition
ss	=	the starting sector of the sub-partition
sl	=	the low byte of the sub-partition size in sectors
sh	=	the high byte of the sub-partition size in sectors

Example:

```
OPEN15,16,15:PRINT#15,"/4:SUB1,"CHR$(1)CHR$(0)
CHR$(160)CHR$(0)",C":CLOSE15
```

The preceding example should be entered as one line. There is no JiffyDOS equivalent for this command since it requires the use of the CHR$ function.

Note: Before you can use a newly-created 1581 sub-partition, you must format it. See 'Formatting 1581 Style Sub-Partitions' for more information.

Deleting Partitions

Partitions are deleted by using the RAM-TOOLS program supplied with RAMLink. Use of this program is documented in Appendix A.

Deleting 1581 Style Sub-partitions

1581 style sub-partitions are handled quite differently than standard RAMLink partitions. You can delete a sub-partition by using the DOS or BASIC 7.0 SCRATCH commands. Scratching a 1581 sub-partition is no different than scratching a file, although the consequences may be severe if this is done accidentally. Any files contained within the sub-partition will be lost. See 'Scratching (deleting) Files' for the proper command syntax. Remember to substitute the name of the sub-partition for the filename.

Changing Partitions

You may change from one partition to another by sending the 'CP' (Change Partition) command to RAMLink via the command channel. The syntax is:

```
OPENlf,dv,15:PRINT#lf,"CPn":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the device number of RAMLink
n = the partition number you wish to change to (1-31)

Example:

```
OPEN15,16,15:PRINT#15,"CP4":CLOSE15
```

JiffyDOS Example:

```
@CP11
```

The 'C'<SHIFT>'P' command is a variation on the Change Partition command which allows it to be used more easily from within BASIC programs. A shifted 'P' is indicated by the symbol '□'. This command allows you to use a character string to indicate the partition. The syntax is:

```
OPENlf,dv,15:PRINT#lf,"C□"+CHR$(n):CLOSElf
```

where: lf = the logical file number for the command channel
dv = the device number of RAMLink
n = the partition number you wish to change to (1-31)

Example:

```
OPEN15,16,15:PRINT#15,"C□"CHR$(11):CLOSE15
```

Moving Between 1581 Style Sub-partitions

Since the 1581 Emulation Mode partitions on RAMLink support 1581 style partitioning (that is to say 'sub-partitioning'), you may use the standard DOS commands to change from one 'sub-partition' to another. The syntax is:

```
OPENlf,dv,15:PRINT#lf,"/[n]:[partname]":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the device number of RAMLink
n = the partition number of the 1581 Emulation partition
partname = the partition name of the 1581 style 'sub-partition'

If the 1581 Emulation Mode partition is the current partition, the partition number n may be left out. The 'sub-partitions' may be nested within each other, but to access one which is two levels down from the currently selected one, you will have to issue the command twice (once with each of the two sub-partition names). To return to the root (main) directory of the 1581 Emulation Mode partition, issue this command without the sub-partition name. This will also occur if you issue an INITIALIZE command to the 1581 Emulation Mode partition.

Please note that if you exit a 1581 Emulation Mode partition with a 'CP' command and then return to it later with another 'CP' command, you will be placed into whichever sub-partition you were in when you exited.

Example of moving to a different 1581 sub-partition:

```
OPEN15,16,15:PRINT#15,"/4:SUB1":CLOSE15
```

JiffyDOS example:

```
@/4:SUB1
```

Formatting Partitions

The standard Commodore DOS NEW command (not to be confused with the BASIC NEW command) may be used to format partitions on RAMLink from either BASIC 2.0 or BASIC 7.0. Although RAMLink will also accept the BASIC 7.0 HEADER command, this command is limited to either the current partition (0) or partition 1.

The DOS NEW commands may be used to delete all files from a partition. It is not necessary to format any partitions on RAMLink before you begin using them, as they have been pre-formatted at the time of creation. Even when you create new partitions on the system, formatting is performed automatically by RAM-TOOLS. You may wish to format them again, however, to change the header name or disk ID. When using the DOS NEW command, RAMLink can accept both the long and short versions. The

difference in time is slight, especially in comparison to other disk drives (formatting rarely takes longer than a second or two). The BASIC 2.0 or BASIC 7.0 syntax for the DOS NEW command is given below.

```
OPENlf,dv,15:PRINT#lf,"N[n]:partname[,id]"
:CLOSElf
```

where:
	lf	=	the logical file number for the command channel
	dv	=	the current device number of RAMLink
	n	=	the partition you wish to format (0-31)
	partname	=	the name you wish to appear in the partition header
	id	=	a two character id for the partition header

The following syntax applies to the BASIC 7.0 HEADER command:

```
HEADER"partname"[,Iid][,Dn][{ON|,}Udv]
```

where:
	partname	=	the name you wish to appear in the partition header
	id	=	a two character id for the partition header
	n	=	the partition number you wish to format (0 or 1)
	dv	=	the device number of RAMLink

Examples:

```
OPEN15,16,15:PRINT#15,"N3:PARTITION 3,P3":CLOSE15
HEADER"PARTITION 1",IP1,D1 ON U16
```

JiffyDOS examples for using the DOS NEW command:

```
@N3:PARTITION 3,P3
@"N3:PARTITION 3,P3",16
```

Note: The DOS NEW and BASIC 7.0 HEADER commands will not be accepted if issued from within a Native Mode subdirectory.

Formatting 1581 Style Sub-partitions

The DOS NEW commands are also used to format the 1581 style sub-partitions which may be created within 1581 Emulation Mode partitions. 1581 sub-partitions must be formatted before they can be used. Before attempting to format a sub-partition, you must make sure that the sub-partition is the current sub-partition within the 1581 Emulation Mode partition that contains it. Do this by using the command outlined in 'Moving Between 1581 Style Sub-partitions' elsewhere in this section. To avoid formatting the wrong area on RAMLink, it is usually wise to make the appropriate 1581 Emulation Mode partition the current partition.

Initializing Partitions

The DOS INITIALIZE command is often used when a different disk is inserted into a floppy disk drive. This ensures that the drive updates its buffer with a copy of the BAM on the new disk. This function is performed automatically by RAMLink, but the command has been implemented to retain compatibility. Note that initializing a 1581 Emulation Mode partition causes it to return to the root directory (ala the 1581). The syntax is:

```
OPENlf,dv,15:PRINT#lf,"I[n][:]":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the current device number assigned to RAMLink
n = the partition to be initialized

Example:

```
OPEN15,16,15:PRINT#15,"I3:":CLOSE15
```

JiffyDOS example:

```
@I3:
```

Validating Partitions

The DOS VALIDATE and BASIC 7.0 COLLECT commands check all files in a partition to verify proper allocation of disk space, free any improperly allocated blocks, and delete unclosed (splat '*') files. You should not use these commands on partitions that contain blocks allocated via the BLOCK-ALLOCATE command, or else information may be lost. The VALIDATE command must be used with BASIC 7.0 if you wish to validate a partition other than the current (0) partition or partition 1. The syntax is:

```
OPENlf,dv,15:PRINT#lf,"V[n][:]":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the device number of RAMLink
n = the partition you wish to validate (0-31)

The syntax for the BASIC 7.0 COLLECT command is:

```
COLLECT[,Dn][{ON|,}Udv]
```

where: n = the partition you wish to validate (0 or 1)
dv = the device number of RAMLink

Examples:

```
OPEN15,16,15:PRINT#15,"V2:":CLOSE15
COLLECT,D0,U16
```

JiffyDOS example of the DOS VALIDATE command:

```
@V2
```

Partition directory

Having multiple partitions on RAMLink necessitates having the ability to view a directory of partitions. The partition directory may be viewed while you are working within any partition and relates information concerning the number, name, and type of each partition on RAMLink. This command also contains options that allow you to specify which partitions will be listed. The syntax for this command is as follows:

```
LOAD"$=P[:*][=tp]",dv
```

where: tp = partition type - N = native
4 = 1541
7 = 1571
8 = 1581
dv = the current device number assigned to RAMLink

Examples:

```
LOAD"$=P",16
LOAD"$=P:*",16
LOAD"$=P:*=8",16
```

JiffyDOS examples:

```
@$=P
@"$=P:*=N",16
```

Renaming Partitions

If you reformat a partition with the DOS NEW command, you may wish to change its name in the partition directory as well. In order to do this we have added the 'Rename Partition' command. This command is similar to the DOS command which is used to rename files. The syntax is as follows:

```
OPEN1f,dv,15:PRINT#1f,"R-P:newname=oldname":CLOSE1f
```

where: lf = the logical file number
dv = the current device number of RAMLink
newname = the name you wish to assign to the partition
oldname = the name of the partition in the partition directory

Example:

```
OPEN15,16,15,"R-P:WORK=NATIVE 1":CLOSE15
```

JiffyDOS Examples:

```
@R-P:WORK=NATIVE 1
@"R-P:WORK=NATIVE 1",16
```

Renaming Directory Headers

After placing a number of files within a given partition or subdirectory, you may wish to change the name that appears in the directory header (displayed at the top of the directory listing for that partition or subdirectory). Although this could be done by using the DOS NEW command, all files within that partition would be lost at the same time. In order to allow you to rename a header without having to lose or copy all of your files, RAMLink has a 'Rename Header' command. The syntax is as follows:

```
OPENlf,dv,15:PRINT#lf,"R-H[n] [path]:newname":CLOSElf
```

where: lf = the logical file number
dv = the current device number of RAMLink
n = the partition where the header is to be renamed
path = the subdirectory path
newname = the new name for the specified header

Examples:

```
OPEN15,16,15,"R-H:WORK":CLOSE15
OPEN15,16,15,"R-H3:DOWNLOADS":CLOSE15
OPEN15,16,15,"R-H1//ASSEM/:BUDDY64":CLOSE15
```

JiffyDOS Examples:

```
@R-H:WORK
@"R-H//ASSEM/:BUDDY64",16
```

Getting Partition Information

The DOS 'Get Partition Info' command has been created for the purpose of gathering information about the current or some other specific partition. This command will prove valuable to the programmer whose software must react differently to partitions of various types. The partition number for which the information is requested may be placed into a variable and inserted into the command as a character string. The syntax for the G-P command is:

```
OPENlf,dv,15:PRINT#lf,"G-P"[+CHR$(n)]:CLOSElf
```

where: lf = the logical file number for the command channel
dv = the device number of RAMLink
n = the partition number (0-31)

If the 'G-P' command is sent without the optional character string or a value of 255, the information returned will be for the current partition. A value of 0 requests that the information returned is to be for the system partition. Thirty bytes (0-29) of information concerning the requested partition plus a CHR$(13) are returned over the error channel. The following is a list of this information:

Byte 0	- Partition type	0	- not created
		1	- Native Mode
		2	- 1541 Emulation Mode
		3	- 1571 Emulation Mode
		4	- 1581 Emulation Mode
		7	- Foreign Mode (Direct Access)
		255	- System

Byte 1 - CHR$(0) (reserved)
Byte 2 - Partition number
Bytes 3-18 - Partition name as displayed in the partition directory
Byte 19 - Starting system address of partition (high byte)
Byte 20 - Starting system address of partition (middle byte)
Byte 21 - Starting system address of partition (low byte)
Bytes 22-26 - CHR$(0) (reserved)
Byte 27 - Size of partition (high byte)
Byte 28 - Size of partition (middle byte)
Byte 29 - Size of partition (low byte)
Byte 30 - CHR$(13)

Note: The values returned in bytes 19-21 and 27-29 are specified in 256 byte blocks. Also keep in mind that any currently undefined bytes may later be used for specific purposes.

Important: To avoid problems with reading information from Partition 13, the G-P command should always be sent with a trailing carriage return (CHR$(13)). The BASIC PRINT# statement will do this for you automatically as long as you do not follow it with a trailing semicolon (;).

Autobooting

It is possible to autoboot from RAMLink when it is used with a C128 or 128D in 128 mode. To do so, RAMLink must either be configured as device number 8, or you must issue the BASIC 7.0 BOOT command. You must also make sure that the current partition has a valid boot sector. The boot sector is located at track 1, sector 0 in all partitions, including Native Mode partitions. An interesting benefit in Native Mode partitions is that the boot sector is always allocated. It is therefore never in danger of being overwritten by files, and cannot be freed by the DOS VALIDATE command. The following syntax applies to the BOOT command:

```
BOOT[[Dn]{ON|,}[Udv]]
```

where: n = the partition where the file is located (0 or 1)
dv = the device number currently in use by RAMLink

The partition number for this command may only contain a zero or a one. Zero is used to indicate the current partition, while one indicates partition number 1. This command will not accept any other partition numbers, due

to a limitation in the BASIC 7.0 command parsing routines. The structure of the boot sector is the same as found on standard Commodore disk drives.

Examples:

```
BOOT U16
BOOT D0,U16
```

Subdirectory Commands

Three new DOS commands have been added to allow you to create and remove subdirectories, as well as to change the current directory. Both the Create and Change commands use a similar syntax, while the syntax for the Remove command has been limited to avoid problems. These commands, like the subdirectories themselves, are very similar to those found in MS-DOS.

Creating Native Mode Subdirectories

The 'Make Directory' command allows you to create Native Mode subdirectories. This command allows you to use standard path syntax. Using a path allows you to create a subdirectory in any native mode partition no matter what your current partition or current directory may be. Here is that syntax:

```
OPENlf,dv,15:PRINT#lf,"MD[n][path]:name":CLOSElf
```

where:
- lf = the logical file number for the command channel
- dv = the current device number assigned to RAMLink
- n = the Native Mode partition where the subdirectory is to be created
- path = the path to the subdirectory in which the new subdirectory will be created
- name = the name of the new subdirectory

The above syntax may seem slightly confusing, so here are a few guidelines to help you understand how this syntax works:

1. The name of the subdirectory you wish to create must always be separated from the rest of the command by a colon (:).

2. If you are creating a subdirectory within another subdirectory, you must specify that subdirectory within the path unless it is your current directory.

3. If subdirectories are specified within the path of the command (to the left of the colon), each subdirectory name must fall between slash (/) characters (only 1 slash is needed between subdirectory names).

4. Paths normally start at the current directory. If you want the path to start at the root directory (the main directory in that partition), the path should begin with two slashes.

5. If the subdirectory is to be created in a partition other than the partition in which you are located, place the partition number at the start of the path (in front of any slashes).

The following examples should help clarify these guidelines:

```
OPEN15,16,15:PRINT#15,"MD:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"MD1:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"MD1//:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"MD1//TEMP/:TEMP2":CLOSE15
OPEN15,16,15:PRINT#15,"MD/TEMP/:TEMP2":CLOSE15
```

JiffyDOS Examples:

```
@MD:TEMP
@MD1:TEMP
@MD1//:TEMP
@"MD1//TEMP/:TEMP2"
@"MD/TEMP/:TEMP2",16
```

Moving Between Native Mode Subdirectories

The 'Change Directory' command allows you to move between Native Mode subdirectories. This command employs the same syntax used in the 'Make Directory' command. Using a path allows you to move to a subdirectory anywhere in the currently selected Native Mode partition. This command will also allow you to change the currently selected directory in any other partition, but will not move you into that directory. In order to move to the current directory of a different partition, you must issue a 'Change Partition' command. The 'Change Directory' syntax is:

```
OPENlf,dv,15:PRINT#lf,"CD[n]{[←]|[[path][:]
subname]}":CLOSElf
```

where:
lf	=	the logical file number for the command channel
dv	=	the current device number assigned to RAMLink
n	=	the Native Mode partition where the subdirectory you wish to make the current directory exists
path	=	the subdirectory path leading to the
subname	=	the name of the subdirectory

Note that you can include the back arrow immediately after 'CD[n]' to move backwards one directory (to the PARENT). The back arrow cannot be combined with any subdirectory path information. See the examples below.

It is not required that you include the colon before the subdirectory name, as long as the subdirectory name is preceded by a slash.

Here are some examples of the Change Directory command:

```
OPEN15,16,15:PRINT#15,"CD:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"CD1//:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"CD1//TEMP/:TEMP2":CLOSE15
OPEN15,16,15:PRINT#15,"CD1←":CLOSE15
OPEN15,16,15:PRINT#15,"CD/TEMP/TEMP2":CLOSE15
```

JiffyDOS Examples:

```
@"CD:TEMP",16
@CD1//TEMP
@CD1//TEMP/TEMP2
@CD1←
@CD/TEMP/TEMP2
```

Deleting Native Mode Subdirectories

The 'Remove Directory' command allows you to delete Native Mode subdirectories. This command does not allow the use of paths in order to avoid problems with removing a subdirectory which is a parent of the directory in which you are located. This command will not allow you to delete a subdirectory which contains any files - you must delete these files first by using the DOS SCRATCH or the BASIC 7.0 equivalent. The following syntax applies to the 'Remove Directory' command:

```
OPENlf,dv,15:PRINT#lf,"RD[n]:subname":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the current device number assigned to RAMLink
n = the partition where the subdirectory you wish to remove exists
subname = the name of the subdirectory you wish to remove

Here are some examples of the Remove Directory command:

```
OPEN15,16,15:PRINT#15,"RD3:TEMP":CLOSE15
OPEN15,16,15:PRINT#15,"RD:TEMP2":CLOSE15
```

JiffyDOS Example:

```
@"RD3:TEMP",16
@RD:TEMP2
```

Viewing Directories

Directories may be viewed by using the following BASIC 2.0 command:

```
LOAD "$",16
```

This command will load the current directory from your RAMLink (assuming it is set as device number 16). You may then issue the LIST command to view the directory on your screen. You may also use a partition number for selecting the directory to be loaded, as in the following example:

```
LOAD "$2",16
```

Pattern Matching

Selective directories may also be loaded in the standard way, by placing a colon (:) at the end of the partition number or path, and by using a filename or pattern matching characters to determine which files to include in the listing. The equals (=) sign and a filetype designator may also be included after the filename to indicate a particular filetype. The filetype characters are: P for program (PRG), S for sequential (SEQ), U for user (USR), R for relative (REL), and B for subdirectory branch (DIR).

Examples:

```
LOAD "$2:S*=P",16
```

This example will load a directory of all PRG files (=P) starting with an S (S*) from partition number 2. You may also use the asterisk at the beginning of a filename as in the following example:

```
LOAD "$1/UTILS/:*E=P",16
```

This example will load a directory of all PRG files which end with an 'E'. The question mark "?" may also be used to replace an unknown character in the filename. It is also possible to use the asterisk in the middle of a pattern as shown in this example:

```
LOAD "$1/UTILS/:R*E=P",16
```

This pattern will match filenames like RIDE and RUE. Only one asterisk may be used in the pattern. Another matching character is the question mark which will match any character found at that position in the filename.

```
LOAD "$2:B?RE=P",16
```

This example will load a directory of all PRG files which are four characters long, start with 'B' and end with 'RE'. More than one question mark may be used in a pattern. It is also possible to mix question marks and an asterisk together in a pattern.

File Commands

File commands are the most commonly used commands. They include loading and saving files, verifying, renaming, scratching, copying, and locking files and are entered in much same manner as for Commodore or compatible disk drives. Although the BASIC 2.0 versions of these commands are supported in BASIC 7.0, BASIC 7.0 also contains other commands that perform the same functions. Note that the BASIC 2.0 versions of these commands allow more versatility when dealing with partitions.

Loading Files

The following syntax can be used to load programs in BASIC 2.0 and BASIC 7.0:

```
LOAD"[[n][path]:]filename",dv[,sa]
```

where:	n	=	is any legal partition number from 1 to 31
	path	=	the subdirectory path to the file
	filename	=	is any legal filename of up to 16 characters
	dv	=	is the current device number assigned to RAMLink
	sa	=	is the secondary address if needed

To load a machine language program, a secondary address of 1 must be added to the end of this command, separated from the device number by a comma .

Examples:

```
LOAD "2:BASEBALL",16
LOAD "3/TERMS/:TERMBOOT",16,1
```

JiffyDOS examples:

```
/"2:BASEBALL",16
/2:BASEBALL
%3/TERMS/:TERMBOOT
```

It is generally a good idea to use the BASIC 2.0 syntax if you are specifying partitions since BASIC 7.0 will only allow access to the current (0) partition or partition 1, and will not allow the use of subdirectory paths.

The BASIC 7.0 BLOAD command can be used to load machine language or data files into memory. The DLOAD command is used primarily to load BASIC programs. The syntax for these commands is shown below.

```
BLOAD"filename"[,Dn][{ON|,}Udv][,Bb][,Pa]

DLOAD"filename"[,Dn][{ON|,}Udv]
```

where: filename = the name of the machine language program to be loaded
n = the partition number where the file is located (0 or 1)
dv = the device number currently in use by RAMLink
b = the memory bank where the file is to be loaded
a = the starting address of the file to be loaded

Examples:

```
BLOAD"SPRITE",D0,U9,B0,P3584
DLOAD"TEST",D0 ON U9
DLOAD"TEST2"
```

Saving Files

The following syntax can be used to save programs in BASIC 2.0 and BASIC 7.0:

```
SAVE "[[@][n][path]:]filename",dv
```

where: n = is any legal partition number from 1 to 31
path = the subdirectory path where the file is to be saved
filename = is any legal filename of up to 16 characters
dv = is the current device number assigned to RAMLink

The '@' symbol shown in the command syntax may be used to indicate that a file with the same name which already exists should be replaced with the new file. This is called the 'Save with Replace' option and if it is used, it must be followed by a partition number and a colon (:). To save a machine language program from a C64 or C128 in 64 mode, you must use a machine language monitor or change some of the BASIC pointers.

Examples:

```
SAVE"2:BASEBALL",16
SAVE"/TERMS/:TERMBOOT",16
```

JiffyDOS examples:

```
←"2:BASEBALL",16
←/TERMS/:TERMBOOT
```

You may use the BASIC 7.0 BSAVE and DSAVE commands when it is your intention to work with your current partition (0) and directory, or in the current directory of partition 1. BSAVE is intended for files other than BASIC programs, while DSAVE is intended for BASIC programs.

```
BSAVE"[@]filename"[,Dn][{ON|,}Udv][,Bb],Pa TO Pe

DSAVE"[@]filename"[,Dn][{ON|,}Udv]
```

where: filename = the name of the file to be saved
n = the partition number where the file is to be saved
dv = the device number currently in use by RAMLink
b = the memory bank where the file is to be saved
a = the starting address of the file to be saved
e = the ending address of the file to be saved

The '@' symbol may be included to indicate that the file being saved is to replace an existing file with the same. This is called 'Save with Replace'.

Examples:

```
BSAVE"SPRITE",D0,U9,B0,P3584
DSAVE"TEST",D1 ON U9
DSAVE"TEST2"
```

Verifying Files

BASIC 2.0 and 7.0 contain commands which allow you to verify if a program has been saved properly. These commands compare the saved program with the contents of memory. Keep in mind that any change in the contents of memory may cause a verify operation to fail. It is best to verify a file immediately after saving it for this reason. Both versions of BASIC support specifying a partition within the filename portion of this command (as described earlier). The following syntax applies to these commands:

```
VERIFY"[[n][path]:]filename",dv[,sa]
```

where: n = the partition where the file is located
path = the subdirectory path to the file you wish to verify
filename = the name of the file to be verified
dv = the current device number assigned to RAMLink
sa = secondary address of 1 to verify a non-BASIC file

Examples:

```
VERIFY"NEWSTATS",16
VERIFY"1/UTILS/TERMS/:XLATOR",16
```

JiffyDOS example:

```
'"NEWSTATS",16
```

In BASIC 7.0, the standard VERIFY command is accepted, but you may also use the DVERIFY command. This command is limited to use with the current partition (0) or partition 1, and the current subdirectory .

```
DVERIFY"filename"[,Dn][{ON|,}Udv]
```

where: filename = the name of the file to be verified
n = the partition where the file resides (0 or 1)
dv = the device number of RAMLink

Example:

```
DVERIFY"NEWSTATS",D1,U16
```

Renaming Files and Subdirectories

Filenames and Native Mode subdirectory names may be changed by using either the DOS RENAME or the BASIC 7.0 RENAME command. The BASIC 7.0 version only supports the current directory within the current partition (0) or partition 1. When using either version, the partitions specified for the two file names must either be the same, or must indicate the same partition. The syntax for the DOS RENAME command is:

```
OPENlf,dv,15:PRINT#lf,"R[n][path]:newname=[[n]
[path]:]filename":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the current device number of RAMLink
n = the partition where the file to be renamed is located
path = the subdirectory path to the file you want to rename
newname = the new name to be assigned to the file
filename = the name of the file which is being renamed

Examples:

```
OPEN15,16,15:PRINT#15,"R1:BOOT1=BOOT":CLOSE15
OPEN15,16,15,"R1/UTILS/:NEWT=1/UTILS/:WW":CLOSE15
```

JiffyDOS Example:

```
@"R1:BOOT1=BOOT",16
```

The BASIC 7.0 RENAME command syntax is:

```
RENAME[Dn,]"filename"TO[Dn]"newfile"[,Udv]
```

where: n = the partition where *filename* is located
filename = the name of the file which is being renamed
newfile = the new name to be assigned to the file
dv = the current device number of RAMLink

Scratching (deleting) Files

The standard DOS and BASIC 7.0 SCRATCH commands may be used to delete files from a partition. As with many of the other BASIC 7.0 commands, SCRATCH is only effective with partition numbers 0 (current) and 1. The standard Commodore DOS scratch may be used in place of the BASIC 7.0 version when necessary and with BASIC 2.0. The following command syntax covers the DOS version of this command:

```
OPENlf,dv,15:PRINT#lf,"S[n][path]:filename[,[n]
[path]:]filename...]":CLOSElf
```

where: lf = the logical file number for the command channel
dv = the current device number of RAMLink
n = the partition(s) which hold the file(s) to be scratched
path = the subdirectory path(s) to the file(s)
filename = the name of the file(s)

Multiple files may be scratched with this command which will accept up to five separate filename parameters. Different partitions can be specified with the separate filenames. The filename parameters may also contain wildcards to allow scratching of multiple files within a single partition.

Examples:

```
OPEN15,16,15:PRINT#15,"S1:JUNK,3:C?*.BAS":CLOSE15
OPEN15,16,15,"S1/UTILS/:CO*":CLOSE15
```

JiffyDOS Examples:

```
@"S1:JUNK,3:C?*.BAS",16
@S1/UTILS/:CO*
```

The BASIC 7.0 SCRATCH command syntax is:

```
SCRATCH"filename"[,Dn][{ON|,}Udv]
```

where: filename = the name of the file to be scratched
n = the partition where the file to be scratched resides
dv = the device number of RAMLink

Multiple files may also be scratched with this command by using pattern matching, although it does not allow you to specify multiple file names as does the DOS version. Remember, this command is only valid for use with the current directory in partition numbers 0 (current) or 1.

Copying Files

Copying is an important consideration with any storage device. For this reason RAMLink comes supplied with DOS commands which allows you to copy files between partitions. This function may also be accomplished by using one of the copy programs supplied with RAMLink. Another copy program supplied with RAMLink allows you to copy an entire disk to a similar partition on RAMLink and vice versa. For more information on the copy programs included with RAMLink, see Appendix A.

Copying files between RAMLink and other drives

Files may be copied between RAMLink and other disk drives using a standard file copier. Only generic copiers that do not try to discover the drive type by checking ROM locations will work with RAMLink.

We have included FCOPY with RAMLink to assist in file copying. FCOPY will work with all file types and all drive types including an REU running under RAMDOS.

Since RAMLink has JiffyDOS included, you may use the built-in JiffyDOS file-copier with RAMLink as well. Many of the commercial copy programs will not work with RAMLink because they look at specific memory locations to try to identify the drive type being used, or attempt to write drive specific code into the disk drive to speed up the copy process.

Copying and Combining files between partitions

You may copy files from one partition to another on RAMLink by using the standard Commodore DOS COPY command. This command allows you to place a partition number in front of each of the filenames specified in the command. The syntax for this command is as follows:

```
OPENlf,dv,15:PRINT#lf,"C[n][path]:newfile=[[n]
[path]:]filename[,[n:]filename...]":CLOSElf
```

where:

lf	=	the logical file number for the command channel
dv	=	the current device number of RAMLink
n	=	the partition which holds or is to receive the file
path	=	the subdirectory path(s) where the file(s) to be copied or created is (are) located
newfile	=	the name of the new file being created
filename	=	the name of the file(s) which is (are) being copied

Up to five files may be combined into a single file by using this command, though it is important to note that copying a number of files into a single file is only effective with text files. If you use this command for copying a single file from one file to another file in a different partition, you may use the same filename for both files.

Examples:

```
OPEN15,16,15:PRINT#15,"C1:FCOPY=3:FCOPY":CLOSE15
OPEN15,16,15,"C:FULLSTATS=STAT1,3:STAT3":CLOSE15
OPEN15,16,15,"C2:MCOPY=1/COPIERS/:MCOPY":CLOSE15
```

JiffyDOS Examples:

```
@"C1:FCOPY=3:FCOPY",16
@"C2:MCOPY=1/COPIERS/:MCOPY"
@C:FULLSTATS=STATS1,3:STATS3
```

You may also copy files from one partition to another on RAMLink by using the standard BASIC 7.0 COPY command. This command is limited to copying files in the current directory of the current partition (0) or partition 1. Use the DOS COPY command mentioned earlier if you want to copy files between other partitions. The syntax for this command is as follows:

```
COPY[Dn,]"filename"TO[Dn]"newfile"[,Udv]
```

where:	n	=	the partition which holds or is to receive the file
	filename	=	the name of the file(s) which is being copied
	newfile	=	the name of the file being created
	dv	=	the current device number of RAMLink

If you use this command to copy a file to a different partition, you may use the same filename for both files. If they are to reside in the same partition, you must use different filenames or an error will result.

Two files may be combined into a single file by using the BASIC 7.0 CONCAT command, although it is important to note that adding files together in this manner is only effective with text files. The BASIC 7.0 limitation of using only the current directory of the current partition (0:) or partition 1 applies to this command. The syntax is:

```
CONCAT[Dn,]"filename"TO[Dn]"newfile"[,Udv]
```

where:	n	=	the partition where the file exists
	filename	=	the name of the file which is being added
	newfile	=	the name of the file being added to
	dv	=	the current device number of RAMLink

The file previously named *newfile* will be replaced by the newly created combined file. The only way these files may have the same name is if they exist in different partitions.

Example:

```
CONCAT "NEWNUMBERS" TO "ALLNUMBERS"
```

Locking and Unlocking Files

Files located on RAMLink may be locked to avoid scratching them by accident. Before you can scratch a locked file, it must first be unlocked. Locking a file sets one of the bits in the filetype byte for that file (located in the directory entry of that file). Files which have been locked will appear in the directory with a 'less-than' symbol to the right of the filetype. For example, a file named JIFFYMON which has been locked will appear in the directory listing as:

```
33    "JIFFYMON"          PRG<
```

It is also possible to lock Native Mode subdirectories and 1581 sub-partitions. If a subdirectory has been locked, it is not possible to delete it with the 'Remove Directory' command until it has been unlocked.

The Lock command is a 'toggle' function. Using it on an unlocked file will cause the file to become locked. Using it on a file which has already been locked will unlock that file. The syntax for locking and unlocking files is:

```
OPENlf,dv,15:PRINT#lf,"L[n][path]:name":CLOSElf
```

where:	lf	=	the logical file number
	dv	=	the current device number of RAMLink
	n	=	the partition number in which the file exists
	path	=	the Native Mode subdirectory path in which the file exists
	name	=	the name of the file or subdirectory you wish to lock or unlock

The following examples illustrate the use of this command:

```
OPEN15,16,15:PRINT#15,"L:TEST":CLOSE15
OPEN15,16,15:PRINT#15,"L1//:TEST":CLOSE15
OPEN15,16,15:PRINT#15,"L/UTILS/:TEST":CLOSE15
```

JiffyDOS examples:

```
@L:TEST
@"L1//:TEST"
@"L/UTILS/:TEST",16
```

Note: JiffyDOS contains its own version of the LOCK command. This may also be used with RAMLink. See your JiffyDOS manual for details on using this version.

Relative File Commands

Relative files are files which contain an index table to allow quicker access to a particular portion of the file called a record. Records are kept track of by a special section of the relative file called a side sector. Two different types of side sectors exist in RAMLink: Regular side sectors (the default type used in 1541 and 1571 Emulation Mode partitions), and super side sectors (the default in 1581 Emulation Mode and Native Mode partitions).

Relative files may be up to 720 blocks long when regular side sectors are used. Relative files which use a super side sector may be over 60,000 blocks long. Up to 65,535 records are allowed and each record may be from 2 to 254 characters in length. All records in the same file are the same length, although it is not necessary to use all of the characters in each record.

The main advantage of using a relative file is speed. Normally, you must read from the beginning of a file until the information you wish to find is reached. With relative files, as long as you keep track of which record your information is stored in, you may go directly to that area of the file and read or write the required information right away. A separate index is usually kept in another file type to keep track of the particular information that each record contains.

Opening or Creating a Relative File

The same syntax can be used to create new or open existing relative files by using the BASIC 2.0 OPEN command or the BASIC 7.0 DOPEN command. When creating a new relative file, print a new record to the file after opening it, using the PRINT# command. When you are done accessing the relative file, close it with the CLOSE command, or the DCLOSE command if you used the BASIC 7.0 syntax to open it. Here is the BASIC 2.0 syntax for opening or creating a new relative file:

```
OPEN lf,dv,sa,"[[n][path]:]filename[{"|,L,"
+CHR$(rl)}]
```

where:	lf	=	the logical file number
	dv	=	the current device number of RAMLink
	sa	=	the secondary address (2 through 14)
	n	=	the partition in which the file exists or is to be created
	path	=	the Native Mode subdirectory path which leads to the file
	filename	=	the name of the relative file
	rl	=	the record length (only needed when creating a new relative file)

Examples:

```
OPEN2,16,2,"1/DATA/:CUSTOMERS,L,"+CHR$(127)
OPEN2,16,2,"ADDRESS"
```

This syntax may also be used in BASIC 7.0, but an alternate syntax was provided for this newer version of BASIC. Again, as with all BASIC 7.0 specific commands, you will be limited to using the current partition (0) or partition 1, and you will not be able to specify subdirectories. Here is that alternate syntax:

```
DOPEN#lf,"filename"[,Lrl][,Dn][,Udv]
```

where:
lf	=	the logical file number
filename	=	the name of the relative file
rl	=	the record length (only needed when creating a new relative file)
n	=	the partition in which the file exists or is to be created (only 0 or 1 is accepted)
dv	=	the current device number of RAMLink

Examples:

```
DOPEN#2,"CUSTOMERS",L127,D1,U16
DOPEN#2,"ADDRESS"
```

Positioning to a Specific Record

When you are ready to read or write a specific record, it is necessary to use either the DOS POSITION command, or the BASIC 7.0 RECORD command. The relative file must already be opened in order to use these commands, and in the case of the DOS POSITION command, you must also have the command channel open to RAMLink. These commands may also be used to create a number of blank records when first creating a relative file. This makes writing the actual data much faster. Here is the syntax for the DOS POSITION command:

```
PRINT#lf,"P"+CHR$(ch)+CHR$(lr)+CHR$(lh)[+CHR$(of)]
```

where:
lf	=	the logical file number for the command channel
ch	=	the secondary address used when opening the relative file plus a value of 96
lr	=	the low byte of the record number you wish to access or create
lh	=	the high byte of the record number you wish to access or create
of	=	the byte number in the record which you wish to start reading from or writing to (first byte if left out)

Examples:

```
PRINT#15,"P"+CHR$(98)+CHR$(30)+CHR$(0)+CHR$(10)
PRINT#15,"P"+CHR$(98)+CHR$(30)+CHR$(0)
```

Here is the syntax for the BASIC 7.0 RECORD command:

```
RECORD#lf,rn[,of]
```

where: lf = the logical file number for the relative file
rn = the record number you wish to access or create
of = the byte number in the record which you wish to start reading from or writing to (first byte if left out)

Examples:

```
RECORD#2,30,10
RECORD#2,30
```

If you are creating a relative file ahead of time, use the PRINT# command to print a CHR$(255) to the last record. This character has a special meaning in relative files - it is used as the first character for empty records. You may expand the size of the relative file later if you run out of records simply by writing a record with a higher number than currently exists in the file. It just takes a little longer to write a record which has not been previously created.

Whenever a new record is created, an error will occur in RAMLink. This is error number 50 which means "RECORD NOT PRESENT". Obviously, if it was your intent to create this record, this error can be ignored.

Note: Although it is not necessary to send the RECORD or POSITION commands twice on RAMLink to avoid data corruption, this practice should be followed anyway to avoid problems when using other drives with your program. To do this, send the RECORD or POSITION command once before writing a record as you normally would, and once again afterward. This will help to ensure that your data will not be corrupted due to the flaw which exists in other disk drives.

Special RAMLink Commands

Some additional commands have been provided in RL DOS to allow you to take advantage of special features built into this unique system.

Software SWAP Commands

RAMLink allows you to perform the SWAP 8 and SWAP 9 functions from within software in addition to using the front panel switches. You may also undo any currently active SWAP condition with a SWAP TO DEFAULT command. These commands have been included to allow loaders or boot programs to easily swap RAMLink's device number.

Whenever one of the SWAP commands is used within a program, it is very important to include a delay loop after issuing the command, even before you attempt to close the command channel. Since these commands instruct RAMLink to send commands to another drive on the system via the serial bus, any attempt by the computer to use the serial bus while this is taking place could cause a bus collision. Although we have found that delay values of 60 or higher within a FOR/NEXT loop will work with most systems, it is recommended that you use a larger value (500 would be good) to leave ample time for these commands to complete their tasks.

Note that the SWAP commands will not operate under certain conditions, usually when there is a file open to a serial bus device In this case, it would be unwise to perform a SWAP anyway. To avoid these kinds of problems, make sure that you send the SWAP command only when no other files (with the exception of the command channel) are open.

Sending a SWAP TO DEFAULT command is not required when you wish to change from a SWAP 8 condition to a SWAP 9 condition, or vice-versa (RAMLink will swap directly from 8 to 9 or from 9 to 8). SWAP TO DEFAULT is only needed when you wish to return RAMLink to it's default device number. It is also possible to send the swap commands from direct mode. You may also abbreviate the command if you are using JiffyDOS. The following syntax applies to the SWAP commands:

```
OPENlf,dv,15:PRINT#lf,"S-x":
FORt=1TO500:NEXT:CLOSElf
```

where:	lf	=	the logical file number
	dv	=	the current device number of RAMLink
	x	=	8 to swap with device number 8 9 to swap with device number 9 D to return RAMLink to its original device number
	t	=	a variable to be used for the timing loop

Here is a sample program using all of the SWAP commands. This program assumes that RAMLink is device number 16 at the time the program begins.

```
100 OPEN15,16,15
110 PRINT#15,"S-8":FORI=1TO500:NEXT:CLOSE15
120 OPEN15,8,15
130 PRINT#15,"S-9":FORI=1TO500:NEXT:CLOSE15
140 OPEN15,9,15
150 PRINT#15,"S-D":FORI=1TO500:NEXT:CLOSE15
```

Direct mode example:

```
OPEN15,16,15,"S-8"
CLOSE15
```

Note: Using these commands in direct mode will not require the timing loop as long as you send the SWAP command first, and then close the channel on a second line. By the time you finish typing the CLOSE command, the SWAP function will be done using the serial bus.

JiffyDOS examples:

```
@S-8
@"S-9",8
```

Parallel Control Commands

In some cases it may be necessary to turn off the parallel port if you are using RAMLink with a CMD HD Series hard drive. A Parallel On/Off command has been added to allow you to do this. This command is similar to JiffyDOS enhanced wedge commands, as it is issued by using the @ symbol. To turn off the parallel port, issue the following command:

```
@P0
```

To turn the parallel port on (default), issue the same command, replacing the 0 (zero) at the end of the command with a 1 (one), like this:

```
@P1
```

Direct Access Commands

Most direct access commands require that files be opened to both the command channel and to a direct access file. Before using the commands described in this section, you should be familiar with the methods required to open and access the command channel and a file data channel.

The Direct Access Channel

Opening a direct access channel requires the following BASIC statement:

```
OPENlf,dv,sa,"#[bu]"
```

where: lf = the logical file number
dv = the device number of RAMLink
sa = the secondary address
bu = the buffer number to be used

The logical file number can be any number from 1 to 127, but cannot be the same any logical file currently in use.

The device number is the one currently assigned to RAMLink (8-29).

The secondary address (also referred to as the channel number) may be any number from 2 to 14. The channel number should be different than any other currently active channel number on RAMLink. It is usually a good practice to use the same number for the logical file number and the channel number. This usually makes keeping track of files easier for the programmer.

The buffer number may currently be any number from 0 through 7. This is an optional parameter, and if left out, RAMLink will automatically assign the next available buffer. It is usually best to leave the buffer number out unless you need to use a specific one. If you select the buffer number yourself, be sure to check for an error, since selecting a buffer which is already in use will produce an error condition. This example show how to open a direct access file:

```
OPEN2,16,2,"#"
```

Note: Direct access files are always opened to the current partition number on RAMLink. If you wish to open a direct access file to a partition other than the one you are currently in, you must change partitions with the 'CP' command first. All further access to this file will occur in that partition number, even if you change partitions after opening the file.

Reading and Writing Data with Direct Access

When BLOCK-READ or BLOCK-WRITE commands are being used, information is normally read or written using the BASIC commands GET#, INPUT#, and PRINT#. The data being accessed with these commands is not being written to or read from the disk directly, but is actually stored in a buffer which temporarily holds the data for a particular block on disk. This may seem a little confusing to those who have never used these commands before, so here is a simpler way of looking at the process.

Reading Data from RAMLink

If you wish to read data directly from the RAM disk, a BLOCK-READ or U1 command is sent to RAMLink to tell it to read a particular block from the RAM disk. This block is placed into a 256-byte RAM buffer in RAMLink. To transfer this data from RAMLink to the computer, the BASIC commands GET# or INPUT# are used. If you only need a portion of the data in the block, you can use the BUFFER-POINTER command to tell RAMLink which byte will be the first to be passed to the computer with GET# or INPUT#.

Here is a quick example of reading the seventh byte from track 1 sector 0 of device number 16. The buffer points to byte 6 because the byte numbers start at 0, so byte 6 is actually the seventh byte.

```
10 OPEN15,16,15:REM OPEN COMMAND CHANNEL
20 OPEN2,16,2,"#":REM DIRECT ACCESS CHANNEL
30 PRINT#15,"U1";2;0;1;0:REM TRACK 1 SECTOR 0
40 PRINT#15,"B-P";2;6:REM SEVENTH BYTE (6)
50 GET#2,A$:REM READ THE BYTE INTO A$
60 CLOSE2:REM CLOSE DIRECT ACCESS CHANNEL
70 CLOSE15:REM CLOSE COMMAND CHANNEL
```

If you know which buffer is being used, a MEMORY-READ command could also be used for this purpose. Here is an example of this method:

```
10 OPEN15,16,15:REM OPEN COMMAND CHANNEL
20 OPEN2,16,2,"#2":REM USE BUFFER #2
30 PRINT#15,"U1";2;0;1;0:REM TRACK 1 SECTOR 0
40 PRINT#15,"M-R"CHR$(6)CHR$(5):REM LOC $0506
50 GET#15,A$:REM READ THE BYTE INTO A$
60 CLOSE2:REM CLOSE DIRECT ACCESS CHANNEL
70 CLOSE15:REM CLOSE COMMAND CHANNEL
```

Writing Data to RAMLink

If you wish to write data directly to the disk, the PRINT# command is first used to write this data to a 256-byte buffer in RAMLink. After the data has been written to this buffer, the buffer itself is then transferred to the desired track and sector location on the RAM disk by using the BLOCK-WRITE or U2 command.

Here is a quick example of writing to the seventh byte on track 1 sector 0 of device number 16. As in the previous example, the buffer points to byte 6 because the byte numbers start at 0, so byte 6 is actually the seventh byte.

```
10 OPEN15,16,15:REM OPEN COMMAND CHANNEL
20 OPEN2,16,2,"#":REM DIRECT ACCESS CHANNEL
30 PRINT#15,"B-P";2;6:REM SEVENTH BYTE (6)
40 PRINT#2,CHR$(65):REM WRITE BYTE TO BUFFER
50 PRINT#15,"U2";2;0;1;0:REM TRACK 1 SECTOR 0
60 CLOSE2:REM CLOSE DIRECT ACCESS CHANNEL
70 CLOSE15:REM CLOSE COMMAND CHANNEL
```

The above example writes a sector to the disk with no regard for what existed in that sector previously. If you want to change a byte in the sector without changing the rest of the contents of that sector, read it into the buffer first, change the desired byte to the new value, and then write the sector back to disk.

If you know which buffer is being used to write the data, a MEMORY-WRITE command might also be used for this purpose. Here is an example of this method:

```
10 OPEN15,16,15:REM OPEN COMMAND CHANNEL
20 OPEN2,16,2,"#2":REM USE BUFFER #2
30 PRINT#15,"M-W"CHR$(6)CHR$(5)CHR$(1)CHR$(65)
:REM LOC $0506, ONE BYTE, VALUE 65
40 PRINT#15,"U2";2;0;1;0:REM TRACK 1 SECTOR 0
50 CLOSE2:REM CLOSE DIRECT ACCESS CHANNEL
60 CLOSE15:REM CLOSE COMMAND CHANNEL
```

Again, this example writes the new sector to the disk without regard to what existed in the sector previously. If you wish to change a byte without changing the rest of the contents in a sector, read the sector into the buffer first, change the byte, and then re-write it to the disk.

Block Commands

The BLOCK commands allow you to read, write, allocate, and de-allocate the sectors on the disk. When using these commands, you should keep in mind which type of partition you are working with, since each partition has different legal track and sector values, with the directory, BAM, and header

information stored in different areas. Many of the functions performed with the BLOCK commands may also be performed with the USER commands, and in fact, the USER commands are often preferred since they do not exhibit some of the side effects of the BLOCK commands.

Allocating Blocks

The BLOCK-ALLOCATE command is used to allocate a block directly. When saving or writing files, this function is handled automatically by the DOS. BLOCK-ALLOCATE is normally used after information has been written to disk by a BLOCK-WRITE command. It can also be used to determine the next higher unallocated block available, if you try to allocate a block which has already been allocated. Upon reading the error channel, you will find that the track and sector variables of the error message will contain the track and sector of the next available block. The syntax for the BLOCK-ALLOCATE command is:

```
PRINT#lf,"B-A:";n;t;s
```

where:
- lf = the logical file number used for the command channel
- n = the partition (0) in which the block to be allocated exists
- t = the track on which the block to be allocated exists
- s = the sector of the block to be allocated

Important Note: The bugs that existed with some Commodore drives with the BLOCK-ALLOCATE command do not exist with the RAMLink version of this command. There were two different bugs associated with this command. One of these caused the entire track (instead of just one sector) to be allocated on 1541 and 1571 disk drive units. The other bug caused the BAM on 1581 disk drives to be improperly allocated if another block command (such as B-W) was issued after a block allocate. Closing the file or issuing another type of disk command usually got around this problem, but these steps are unnecessary when using RAMLink.

Freeing Blocks

The BLOCK-FREE command allows you to free a block which is currently allocated. When scratching files, this function is handled automatically by the DOS. BLOCK-FREE is normally used to free a block in which information had been previously placed by a BLOCK-WRITE command, but which is no longer needed. The syntax for the BLOCK-FREE command is:

```
PRINT#lf,"B-F:";n;t;s
```

where:
- lf = the logical file number used for the command channel
- n = the partition (0) in which the block to be allocated exists
- t = the track on which the block to be allocated exists
- s = the sector of the block to be allocated

The Buffer Pointer

The BUFFER-POINTER command allows you to point to a specific byte within a sector which is to be read from or written to. This action occurs within the disk buffer which holds the data for the particular. The syntax for this command is:

```
PRINT#lf,"B-P";ch;pt
```

where: lf = the logical file number used for the command channel
ch = the channel number used for the direct access file
pt = the byte number within the block you wish to be used as the first byte of the next GET#, INPUT#, or PRINT# statement

The syntax given above assumes that the command and direct access channels have already been opened. The channel number is the same as the secondary address you used when opening the direct access channel. When using the BUFFER-POINTER command, remember that bytes are numbered from 0 to 255. If you need to access the first byte, it is byte 0.

Reading Blocks

The BLOCK-READ command, due to a quirk in the way it operates, is rarely used. Instead, most applications use the 'U1' command, which is very similar to the BLOCK-READ command. The syntax is:

```
PRINT#lf,"B-R";ch;n;t;s
```

where: lf = the logical file number used for the command channel
ch = the channel number used for the direct access file
n = partition number (always 0)
t = the track from which you wish to read a block
s = the sector which you wish to read from

The channel number specified above is the same as the secondary address used to open the direct access file.

The partition number should always be 0 (zero). This is because direct access commands will always access the partition that was the current partition at the time the direct access channel was opened.

The problem with the BLOCK-READ command is that it will not read an entire block as data. Instead, it uses the first byte in the block as a counter for how many bytes are to be read. This allows you to read the full block only if the first byte of the block contains a value of 255. Since most blocks contain a link to the next sector in the first byte, any blocks intended to be read in this fashion must be written with this same consideration in mind.

Writing Blocks

The BLOCK-WRITE command, due to a quirk in the way it operates, is rarely used. Instead, most applications use the 'U2' command, which is nearly identical to the BLOCK-WRITE command. The syntax for the BLOCK-WRITE command is:

```
PRINT#lf,"B-W";ch;n;t;s
```

where:
- lf = the logical file number used for the command channel
- ch = the channel number used for the direct access file
- n = partition number (always 0)
- t = the track to which you wish to write a block
- s = the sector which you wish to write to

The channel number specified above is the same as the secondary address used to open the direct access file.

The partition number should always be 0 (zero). This is because direct access commands will always access the partition that was the current partition at the time the direct access channel was opened.

The problem with the BLOCK-WRITE command is that it will not write an entire block as data. Instead, it uses the first byte in the block as a counter for how many bytes are to be written. This will allow you to write (and later read) the full block only if the first byte of the block contains a value of 255.

Block Execute

The BLOCK-EXECUTE command has not been fully implemented in RAMLink. If RAMLink receives a BLOCK-EXECUTE command, it will not create and error, but will simply return without executing the command.

Memory Commands

The memory commands allow you to read from or write to RAMLink system memory. The memory commands are useful for many applications, but should be handled carefully to avoid corrupting data or important locations used by the DOS.

Reading from RAMLink System Memory

The MEMORY-READ command allows you to read directly from system memory locations within RAMLink. This command is similar to the BASIC PEEK command, with the difference being that MEMORY-READ returns the contents of a serial device's memory instead of computer memory. Reading memory is useful in many applications such as

determining the type of device being used as a particular device number. The syntax for MEMORY-READ is:

```
PRINT#lf,"M-R"CHR$(ml)CHR$(mh)CHR$(nb)
```

where:
	lf	=	the logical file number used for the command channel
	ml	=	the low byte of the starting memory address
	mh	=	the high byte of the starting memory address
	nb	=	the number of bytes to be read

The value for the number of bytes to be read can be from 0 to 255. Since it is improbable that you would want to read zero bytes from RAMLink's system memory, and it is also likely that you may wish to read 256 bytes on many occasions, the value of zero has been altered to mean 256.

After a MEMORY-READ command is sent, the specified bytes are returned over the error channel. Thus, you may use the GET# command to read these bytes one at a time.

To determine the low and high bytes of a decimal address, you can use the following formula:

```
HB=INT(AD/256):LB=AD-HB*256
```

where:
	AD	=	the decimal address you wish to begin reading from
	HB	=	the high byte of the starting memory address
	LB	=	the low byte of the starting memory address

Here is an example program which determines if the device being accessed is a RAMLink unit:

```
10 OPEN15,16,15
20 PRINT#15,"M-R"CHR$(160)CHR$(254)CHR$(6)
30 FORI=1TO6:GET#15,B$:A$=A$+B$:NEXT
70 CLOSE15
80 IFA$="CMD RL"THENPRINT"RAMLINK PRESENT":END
90 PRINT"NOT A RAMLINK UNIT":END
```

Note: The MEMORY-READ command will not read past a page boundary.

Writing to RAMLink System Memory

The MEMORY-WRITE command allows you to write to memory locations within RAMLink. Using this command is similar to using the POKE command in BASIC, the difference being that MEMORY-WRITE writes to RAMLink system memory instead of computer memory. MEMORY-WRITE is useful for placing data directly into one of the buffers, the job queue, or other significant memory locations within RAMLink. The syntax for MEMORY-WRITE is:

```
PRINT#lf,"M-W"CHR$(ml)CHR$(mh)CHR$(nb)CHR$(d)...
```

where:
- lf = the logical file number used for the command channel
- ml = the low byte of the starting memory address
- mh = the high byte of the starting memory address
- nb = the number of bytes to be written
- d = value of the data byte to be written (if more than one byte is to be written use additional CHR$ statements or a string variable)

The value for the number of bytes to be written can range from 1 to 127. When sending a MEMORY-WRITE command, the bytes to be written are placed at the end of the command. This may be done in the form of CHR$ statements, or if more room is needed, as a string variable.

To determine the low and high bytes of a decimal address, you can use the formula given in the MEMORY-READ command above.

Here is an example program which illustrates the use of the MEMORY-WRITE command:

```
10 FORI=1TO127
20 A$=A$+CHR$(I)
30 NEXT
40 OPEN15,16,15
50 PRINT#15,"M-W"CHR$(0)CHR$(8)CHR$(127)A$
60 CLOSE15
70 END
```

Memory Execute

The MEMORY-EXECUTE command has not been fully implemented in RAMLink. Sending this command to RAMLink will not cause an error to occur, RAMLink will simply return without performing any action at all.

User Commands

The USER commands provide useful replacements for BLOCK-READ and BLOCK-WRITE and allow you to perform soft reset and checksum functions. Not all of the USER commands have been fully emulated on RAMLink. Those commands which are considered to be burst commands have not been included, and the USER commands normally used to perform machine language jumps into certain buffer locations have been disabled. All USER commands are sent via the DOS command channel (secondary address of 15).

U0 Utility Commands

The U0 command has a number of uses. Without any parameters, it resets the user vectors. When combined with specific parameters, U0 provides a way to send burst and utility commands. Here is the syntax:

```
PRINT#lf,"U0[+|-|>[ut]]"[+CHR$(d)]
```

where:
- lf = the logical file number used for the command channel
- ut = utility command character (if needed)
- d1 = burst utility data byte (if needed)
- d2 = second burst utility data byte (if needed)

Since most of these commands are covered in greater detail in the paragraph titled 'CHGUTL Utility' in the area describing burst commands, we will present only the general syntax of each command here. Assume that each the command begins with a 'PRINT#*lf*,' and that the *lf* should be replaced with an already-opened logical file number for the command channel.

"U0"	*Reset user vectors to default
"U0>"+CHR$(d)	Change device number (d = dev.#)
"U0>C"	Test RAM checksum
"U0>T"	*Test ROM checksum

*Denotes commands which are accepted but do not perform any function.

Note: These commands can be sent with the JiffyDOS wedge with the exception of the command used to change the device number. Commands which require the addition of a character string code to the command cannot be properly parsed by the JiffyDOS wedge.

JiffyDOS Examples:

```
@U0>C
@"U0>T"
@"U0",16
```

Reading Blocks with U1

The U1 command is used for reading specific blocks within a partition. U1 should be considered a direct access command, as is the BLOCK-READ command which U1 is normally used in place of. The U1 command will read an entire block into the buffer which is being used by the direct access channel. The syntax for the U1 command is:

```
PRINT#lf,"U1";ch;n;t;s
```

where:
- lf = the logical file number used for the command channel
- ch = the channel number used for the direct access file
- n = partition number (always 0)
- t = the track from which you wish to read a block
- s = the sector which you wish to read from

The channel number is the same as the secondary address used when opening the direct access file.

The partition number should always be 0 (zero). This is because direct access commands will always access the partition that was the current partition at the time the direct access channel was opened.

After using the U1 command to read a block, you may retrieve the data it stores in the buffer with the GET# command. You may position the pointer with the BUFFER-POINTER command to retrieve specific bytes before using the GET# command. You may also substitute the characters UA for U1. An example of this command can be found under the heading 'Reading and Writing Data with Direct Access' earlier in this section.

Writing Blocks with U2

The U2 command is used for writing specific blocks to a partition. U2 is considered to be a direct access command, as is the BLOCK-WRITE command which U2 is normally used in place of. The U2 command will write an entire block from the buffer which is being used by the direct access channel. The syntax for the U2 command is:

```
PRINT#lf,"U2";ch;n;t;s
```

where:

lf	=	the logical file number used for the command channel
ch	=	the channel number used for the direct access file
n	=	partition number (always 0)
t	=	the track to which you wish to write a block
s	=	the sector which you wish to write to

The channel number is the same as the secondary address used when opening the direct access file.

The partition number should always be 0 (zero). This is because direct access commands will always access the partition that was the current partition at the time the direct access channel was opened.

Before using the U2 command, you should place the data you wish to write into the direct access channel's buffer. This may be accomplished by using the PRINT# command. You may place data in specific bytes by using the BUFFER-POINTER command before sending data with PRINT#. You may also substitute the characters UB for U2. An example of this command can be found under the heading 'Reading and Writing Data with Direct Access' earlier in this section.

User Jump Commands

The majority of the remaining USER commands perform jumps to certain routines in drive memory. Most drives support eight USER JUMP commands in all, but RAMLink supports only two of these. The first six commands, U3 through U8 (alias UC through UH) were normally used to perform jumps to user installed machine language programs located in a drive. Since RAMLink does not contain its own microprocessor, and because routines normally placed into a drive are usually device specific, we felt that these commands would not enhance the performance or compatibility of RAMLink. The remaining two USER commands are used to reset the drive to various degrees, and these have been incorporated into RAMLink. Here are their definitions:

COMMAND	ALIAS	DEFINITION
U9	UI	Warm reset (minimal effect on drive variables)
U:	UJ	Cold reset (does not change current partition)

The syntax for the preceding USER commands is:

```
PRINT#lf,"Ux"
```

where: lf = the logical file number used for the command channel
x = the character of the desired USER command

Burst Commands

Since RAMLink does not connect directly to the serial bus, it is not possible to use most of the common burst commands found on the 1571 and 1581 disk drives. In most cases, burst commands are used to provide high speed data transfer. As RAMLink already provides faster transfer rates than is possible with any serial device, burst commands are not necessary.

Special Loaders

The Commodore DOS Utility Loader and Autoboot Loader have not been implemented into the DOS on RAMLink. If you require automatic execution of certain programs, use RAMLink's capability to autoboot a specific file to accomplish this.

Job Queue Instructions

The job queue buffers are special locations used to pass commands and parameters to RAMLink for the purpose of performing specific RAM disk access functions. The job queue can be accessed directly by the programmer if desired. Different areas are defined for job queues in RAMLink depending on which partition type is currently being used. This was done so that RAMLink could emulate the job queue locations of the 1541, 1571, and 1581. In all cases, the job codes and parameters are moved to the Native Mode job queue for actual execution. When writing job queue routines for RAMLink, it is better to use the Native Mode job queue locations, since the routines will then operate no matter which type of partition is in use.

Job Queue Command Codes

Code	Name	Description
$80	READ	Reads logical block using track and sector parameters
$90	WRITE	Writes logical block using track and sector parameters
$92	DSK_IN_DRV	Not implemented (returns OK status)
$94	ACTIV_ON	Turns ACTIVITY LED on
$96	ACTIV_OFF	Turns ACTIVITY LED off
$98	ERR_ON	Turns ERROR LED on
$9A	ERR_OFF	Turns ERROR LED off
$A0	VERIFY	Not implemented (returns OK)
$B0	SEEK	Not implemented (returns OK)
$C0	BUMP	Not implemented (returns OK)
$D0	JUMP	Not implemented (returns OK)
$E0	EXEC	Not implemented (returns OK)
$F0	FRMT_PART	Not implemented (returns OK)

Job Queue Locations

Address Range	Description
$0000 - $0004	1541/1571 Emulation Mode Job Queue
$0006 - $000F	1541/1571 Emulation Mode Track & Sector
$0002 - $000A	1581 Emulation Mode Job Queue
$000B - $001C	1581 Emulation Mode Track & Sector
$0020 - $0027	Native Mode Job Queue
$0028 - $0037	Native Mode Track & Sector

Direct RAM Access

Direct access to any memory contained within RAMLink may be performed through a set of routines and registers. The register arrangement is similar to the register arrangement used in Commodore REUs, though the base address for these registers differs. This should make it much easier to convert software originally intended for the 17xx series of REUs to utilize RAMLink. Bear in mind that using this type of access to write data directly into the RAM is likely to corrupt data stored in RAM by RL DOS routines.

Direct RAM Access Routines

There are two direct ways of transferring or swapping memory contents between RAMLink and the host computer. The first method uses registers in a manner which is similar to that used by Commodore REUs. The second method allows you to transfer a block of data directly between the computer and RAMLink by using partition, track and sector information to specify the memory areas to be moved in RAMLink.

REU Style Memory Moves

The main difference is that the Commodore REU uses an interrupt based scheme to initiate this type of transfer, while RAMLink requires the controlling program to call an execute routine. RAMLink also requires some other calls in order to make the RAMLink hardware visable to the system, and to set up the REC Image Page registers. The following procedure outlines how to perform an REU type direct memory transfer:

1. Switch in RL Hardware ($E0A9 or $E0B1 shuts off interrupts)
2. Set REC Image Page ($FE03)
3. Set Registers ($DE00 - $DE0E)
4. Call execute ($FE06 or $FE1E)
5. Switch out RAMLink Hardware ($FE0C will turn inturrupts back on or $FE0F to exit without turning on interrupts)

> Note: You may skip step 2 if you use $E0A9 for step 1, and you may skip step 5 if you use $FE1E for step 4.

Sector Addressed Memory Moves

The other method available for direct transfer of memory is intended for moving single 256 byte blocks at a time. These types of transfers use track and sector specifications to identify the RAMLink memory to be transferred. This is similar to a block read or block write operation, but instead of having to read or write each individual block (as is the case with BLOCK-READ and BLOCK-WRITE), the entire block contents are automatically transferred between RAMLink and the host computer's memory. The following procedure outlines the use of this method of transfer:

1. Switch in RL Hardware ($E0A9 or $E0B1 shuts off interrupts)
2. Set REC Image Page ($FE03)
3. Set Registers ($DE20 - $DE25)
4. Call execute ($FE09 or $FE21)
5. Switch out RAMLink Hardware ($FE0C will turn inturrupts back on or $FE0F to exit without turning on interrupts)

Note: You may skip step 2 if you use $E0A9 for step 1, and you may skip step 5 if you use $FE21 for step 4.

Computer Memory Usage

It is important to remember that there are a few computer memory locations which are used by both of the methods outlined above. Four addresses in zero page are used, $00AE through $00B1. The contents of these four locations are saved when the Direct RAM Access execute routines are called, and are restored when the transfer is complete. This should have no effect upon the actual transfer being performed as long as you are not moving memory into zeor page of the computer. If you are moving memory into zero page in the compter, then it will be necessary for you to place any required data into these four locations by using some alternate method.

Processor Stack Usage

The Direct RAM Access routines also use some of the processor stack area during transfers. Four bytes will be pushed on the stack when performing transfers in 64 mode, and five bytes are used for 128 mode transfers. All bytes placed on the stack by the transfer routines are removed from the stack at the end of the transfer. Again, this should have no detrimental effect on your routine unless you are low on stack memory or are attempting to transfer to stack memory. The latter is not recommended.

Direct RAM Access Jump Table

Normally, the RAMLink Direct RAM Access jump vectors and REC Image Registers do not exist in memory. Only by performing a SYS or JSR to routines which enable these vectors and registers will you be able to perform direct access to all RAM within RAMLink. The following is a brief description of the enable routines and assosciated vectors.

$E0A9	57513	EN_SET_REC	Routine to switch in RAMLink Hardware, shut off interrupts, and set REC Image Page.
$E0B1	57521	RL_HW_EN	Routine to switch in RAMLink Hardware and shuts off interrupts.
$FE03	65027	SET_REC_IMG	Routine to set REC Image Page

$FE06	(65030)	EXEC_REC_REU	Executes according to REU register settings
$FE09	(65033)	EXEC_REC_SEC	Executes according to sector move register settings
$FE0C	(65036)	RL_HW_DIS	Disables RAMLink hardware and turns inturrupts back on.
$FE0F	(65039)	RL_HW_DIS2	Disables RAMLink hardware and leaves inturrupts off.
$FE1E	(65054)	EXEC_REU_DIS	Executes according to REU register settings, disables RAMLink hardware and turns inturrupts back on.
$FE21	(65057)	EXEC_SEC_DIS	Executes according to sector register settings, disables RAMLink hardware and turns inturrupts back on.

Register Parameter Descriptions

The following descriptions should assist you in programming your own routines to use RAMLink's Direct RAM Access routines. For some examples of using these routines and registers, you may wish to examine the RAM-TOOLS program, which makes use of both transfer methods.

$DE00 Status Register

Bits 0-3	Unused (always 0)
Bit 4	SIZE (always 1)
Bit 5	VERIFY ERROR (1=error / 0=OK)
Bit 6	TRANSFER COMPLETE
Bit 7	IRQ PENDING (always 0 - IRQ's not used in this scheme)

$DE01 Command Register

Bits 1 and 0	TRANSFER TYPE 00 Move memory from comp. to RL 01 Move memory from RL to comp. 10 Swap computer with RAMLink 11 Verify computer RAM with RAMLink
Bits 2 and 3	Reserved (no specific purpose)
Bit 4	$FF00 Decode (not implemented)
Bit 5	AutoLoad option - resets computer address, expansion address and transfer length to original values specified at beginning of tranfer.
Bit 6	Reserved (No function)
Bit 7	Execute bit (No function)

$DE02-03 Computer Address Pointer

These two bytes form a pointer for the address in the computer which is to be used for memory transfers. Low byte goes in $DE02, high byte in $DE03.

$DE04-06 RAMLink System Address Pointer

These three bytes form a pointer for the address in RAMLink which is to be used for memory transfers. Low byte goes in $DE04, middle byte in $DE05, and the high byte in $DE06.

$DE07-08 Transfer Length

This byte is used to indicate how many bytes are to be transferred. Legal values are from 0-65535. Zero indicates that 65536 bytes are to be transferred.

$DE0A Address Control

This register is used to indicate how addresses should be incremented after the transfer occurs. Only bits 6 and 7 are significant.

Bits 6 and 7 Address Control

- 00 Increment both addresses (the computers address and RAMLink address after you do your transfer)
- 01 RAMLink address fixed
- 10 Computer address fixed
- 11 Both addresses fixed

New Registers in RAMLink

$DE0E Transfer Error

Bit 7 of this register will be set to a value of 1 if an illegal block error occurs while the execute routine is operating. If no illegal block error was encountered, bit 7 will be cleared (set to 0). Errors also effect the carry flag, which is cleared when the RAMLink block was valid, or set if it was invalid.

$DE10 Bank In 128 for Memory Transfer (128 mode)

Indicates which memory bank in the 128 is to be used for memory transfers. Bank numbers used for this location are the same as are used by BASIC 7.0 BANK command.

$DE20 JOB

This register is used to indicate the type of transfer desired. It is also used for reading back errors after the transfer is complete. The following is a list of acceptable job codes:

$80	READ
$90	WRITE
$A0	VERIFY
$B0	SWAP)

ERRORS:	$02	VERIFY ERROR
	$04	ILLEGAL BLOCK
	$0F	ILLEGAL PARTITION

$DE21 Track

This register is used to indicate the track number on RAMLink where the block with which the transfer is to be performed is located.

$DE22 Sector

This register is used to indicate the sector number on RAMLink where the block with which the transfer is to be performed is located.

$DE23-24 Computer Transfer Address

These two bytes form a pointer to the starting address in the computer which will be used for the block transfer.

$DE25 Partition in RAMLink

This register indicates the partition number in RAMLink where the block to be used for the transfer is located. The actual partition number should be used with two exceptions; a zero ($00) is used to indicate that the current partition should be used, and a hex value of $FF indicates that the transfer should occur with the system partition.

$DE10 Bank in 128 for Memory Transfer (128 mode)

Indicates which memory bank in the 128 is to be used for sector transfers. Bank numbers used for this location are the same as are used by BASIC 7.0 BANK command.

REC Image Page and other Registers		
Address		Definition
$DE00	56832	Status Register
$DE01	56833	Command Register
$DE02	56834	Computer Address (low byte)
$DE03	56835	Computer Address (high byte)
$DE04	56836	RAMLink Address (low byte)
$DE05	56837	RAMLink Address (middle byte)
$DE06	56838	RAMLink Address (high byte)
$DE07	56839	Transfer Length (low byte)
$DE08	56840	Transfer Length (high byte)
$DE09	56841	Not used
$DE0A	56842	Address control register
$DE0E	56846	ERROR
$DE10	56848	Bank in 128 for transfer
$DE20	56834	Job
$DE21	56834	RAMLink Track
$DE22	56834	RAMLink Sector
$DE23	56834	Computer Address (low byte)
$DE24	56834	Computer Address (high byte)
$DE25	56834	RAMLink Partition Number
$DE26	56834	Bank in 128 for sector transfer

Appendix A
Utilities

About the Utility Disks

A number of utility programs created by CMD are provided with RAMLink. These programs are located on the floppy disk titled RAMLink Utilities. Be sure to backup this disk immediately. The following is a list of the programs supplied with RAMLink:

RAMLink Utilities

RAM-TOOLS	Partition and configuration utilities
FCOPY	File copier
MCOPY	Whole disk/partition copier
1541SUB	Subdirectory creation utility
1581SUB	Subdirectory creation utility
AUTOFILE EDITOR	Auto-start file configuration editor
AUTO-BOOT 128	Boot-block creation utility for the C128
DISK CRACKER HD	Track and sector editor and drive monitor
HARDWARE TEST	Checks system for RAMLink compatibility
RAM TEST	Tests all RAM connected to the system
ZAP SYSTEM	Erases current system configurations parameters
REWRITE DOS.64	Utility for C64 to place new DOS on the HD
REWRITE DOS.128	Utility for C128 to place new DOS on the HD

Program Documentation

The various programs on this disk are in many cases discussed in other sections of this manual, while actual documentation is provided in this Appendix. Some programs are public domain utilities that have proven to be useful with the RAMLink. These programs may not be fully documented here, but enough information has been included to allow those familiar with these types of utilities to begin using them. Some of the files located on the RAMLink Utilities disk are machine language modules or data files which are used by the other programs, and require no separate documentation.

RAM-TOOLS

RAM-TOOLS is used to create and delete partitions on RAMLink, or to change RAMLink's special configuration parameters. From 64 or 128 mode, load and run RAM-TOOLS. You will be presented with a menu containing several items. These items may be selected by pressing the appropriate number key shown next to the item on the main menu. A description of each of these items and their use is given below.

Default Device Number

RAMLink's auto-configuration will automatically set the default device number to 16. You may change this default to any number between 8 and 29. To do so, select the CHANGE DEFAULT DEVICE NUMBER option from the main menu. The program will display the current RAMLink default device number, and will ask if you wish to change this default. You may press the 'N" key to return to the main menu. If you press 'Y', then you will be propted to select a new device number .Make your selection with the <+> and <-> keys and then press the <RETURN> key. The program will then ask if you wish to write the new configuration to RAMLink. This change will remain in effect until you either change it again, or RAMLink loses power entirely.

We recommend a default value of 9 if you have only one floppy disk on your system, or device number 10 if two floppy drives are being used (remember - you can always use the SWAP functions to switch RAMLink's device number with devices #8 and #9). You should not set RAMLink's default device number to a value which is normally used by another drive, unless you do not plan to use that drive and RAMLink at the same tiem..

Default Partition Number

This option allows you to change which RAMLink partition will be the active partition when your computer is powered up, or when the computer or RAMLink are reset. Auto-configuration sets this default for partition number 1. The process for setting the default partition is similar to the process for setting the default device number. The program will check to see if the partition you select is legal.

Select the CHANGE DEFAULT PARTITION NUMBER option from the menu. The program displays the current default partition number, and will then prompt you to select a new default. Make your selection with the <+> and <-> keys and press the <RETURN> key.

View Partition Table

This option shows the current status of all partitions. The <+> and <-> keys will step you through the pages in the display, and <RETURN> allows you to exit back to the main menu.

Create a New Partition

This option allows you to add new partitions to the system. The program will automatically default to the next available (unused) partition. This is usually the best choice as it will keep your partitions in order. However, if you wish, you may select any partition not yet in use. Use <+> and <-> to change the partition number, and press the <RETURN> key to accept.

You must then select the partition type. Again, use <+> and <-> to view the available choices, and <RETURN> to accept.

If you have not selected an Emulation mode partition, you will need to specify the partition size. The size is adjustable in increments of 256 blocks using <+> and <-> to select, and <RETURN> to accept.

The last step is to enter a name for the partition (16 characters maximum). Even though the usable characters have been limited in this program, you can later rename the partition with the RL DOS Rename Partition command if necessary. After entering the partition name you will be given the option to create the new partition on RAMLink or to abort the process.

Delete an Old Partition

Upon choosing this option you will be shown the first partition on RAMLink. To select the partition that you wish to delete, use the <+> and <-> keys, and then press <RETURN> to accept. After the partition has been selected, you will be given an opportunity to abort the process before the partition is deleted.

Deleting a partition can take from a few seconds to several minutes depending on where the partition to be deleted is located on RAMLink. Partitions located above (at a higher address than) the partition being deleted are moved down to fill the void left by the deleted partition. This is done in order to avoid the fragmentation of storage space. If you are going to delete a number of partitions, start with the last one created, and then continue in reverse order of creation. This will save a lot of time. If you want to create all new partitions from scratch, it may be faster to remove power from RAMLink, allowing it to create only one or two partitions with .

> WARNING: Deleting partitions will destroy all data within the partitions you are deleting. Do not delete partitions without first backing up any data in those partitions which you wish to keep.

Quit

Allows you to exit from HD-TOOLS. Remember to press the RESET switch on the HD before trying to use it again.

Do's and Don'ts

Do:

- plan ahead. Set up your partitions logically.
- backup important data before doing any system configuration. Chances for data loss are much greater when performing these functions.

Don't:

- press RESET or any other switches on RAMLink while a file is open or there is any RAMLink activity.
- make a regular habit of partitioning. This is something you should do once or twice, and edit only when your needs change.

FCOPY

This program was created by CMD to fill the need for a file copier capable of copying any type of file between any two drives or between any two partitions on our devices. In addition, FCOPY supports Native Mode subdirectories, 1581 sub-partitions, and REU's running under RAMDOS. The use of this program is mostly self-explanatory, but we have included a breakdown of the functions here in order to provide more detail where necessary. This program may also be used to remove files, view directories, and send disk commands.

Set Source Device (F1)

This option selects the disk drive that you wish to copy files *from*. The device number and type of device will be shown on the display. If the drive type is not recognized by the program, question marks will be shown instead of the actual type. In the case of unrecognized third party disk drives, the program will work regardless unless the DOS in the particular drive happens to be highly incompatible with standard Commodore-style DOS commands and file handling procedures.

Set Target Device (F5)

This option selects the disk drive that you wish to copy files *to*. The device number and type of device will be shown on the display.

Set Source Partition (F3)

If the source device is a CMD HD, this option allows you to select the partition from which files are to be copied. The partition number, name and type will be displayed in the source area.

Set Target Partition (F7)

If the target device is a CMD HD, this will allow you to set the partition to which files are to be copied. The partition number, name and type will be displayed in the target area.

Set Source Path (S)

If the source device is a CMD HD and the partition type is Native or 1581 Emulation, or if the source device is an actual 1581, a path for subdirectories or sub-partitions may be entered by the user. Each subdirectory or sub-partition name must be separated by a slash (/).

Set Target Path (T)

Same function as Set Source Path, except that the path is intended for the Target device.

Source/Target DIrectory (A/B)

Allows you to view the directory of the source or target disk. The program will ask for a search pattern so that you may view files which match a given name and/or filetype. For directories read from a CMD HD, the pattern will also allow the selection of files by time and date.

Select Files (F)

This option reads the source directory into the selection buffer after you have entered a search pattern. File information is stored in a dynamic method, so if necessary, you will be able to view from 700 to well over a thousand files. If your directory contains more files than can be viewed, you will need to limit the selection by using a pattern which will match fewer files. Once you have entered the file selection mode, you may select or de-select files by pressing the RETURN key while the arrow is pointing to the file. An asterisk (*) indicates which files have been selected. Pressing the 'T' key allows you to toggle all selections ('T' will select all unselected files, and de-select all selected files). When you have finished, press the back-arrow key (←) to return to the main menu.

Reselect Files (R)

You may, at any time after selecting files, return to the file selection mode without reading the directory. This allows you to change your selections before or after copying files.

Send Disk Command (@)

This option allows you to send a disk command to either the source or the target drive. Important: disk commands do not follow the source or target path, but are instead sent to the current directory. Therefore, it is wise to make sure you are currently in the correct partition, sub-partition, or

subdirectory before sending a command intended for a certain area. When sending commands to a CMD HD, you may include a partition number and path in the disk command.

Begin Copying (C)

This option starts the copying process. Only the files you have selected using the 'F' and/or 'R' options will be copied. When copying is complete, an option allows you to copy the same files to another disk (if your target drive is a floppy disk drive).

Begin Scratching (#)

This option starts scratching the files you selected using the 'F' and/or 'R' options. Please note: files can only be scratched from the source disk! You will be asked "ARE YOU SURE (Y/N)?" before the scratching operation begins.

Exit Program (←)

Allows you to exit from program. If you wish to use the program again, it must be re-loaded.

MCOPY

This program is of the type commonly referred to as a 'whole disk copier'. It can copy an entire disk between two floppy drives of the same type, a floppy drive and similar CMD HD partition , or two similar partition types on the same or separate CMD hard drives. The program will support all Commodore floppy drives, as well as any fully compatible third party drives.

MCOPY is also useful for backing up partitions to a floppy disk or to other partitions on the HD. It is also possible to copy Native Mode partitions to other Native Mode partitions of different sizes. This can be useful when you decide that you need a larger partition to hold data. Although it is also possible to copy from a larger partition to a smaller one, it is possible that some data will be lost in this process.

MCOPY is self-documenting. A help menu is always on screen and lists the available options.

1541SUB and 1581SUB

These utilities are nearly identical in function and are used to create Native Mode subdirectories which emulate the directories of 1541 and 1581 drives. Using these types of subdirectories may allow the use of programs which will not normally work in a Native Mode partition.

WARNING: Use these utilities on an empty Native Mode partition only. Any data stored in the partition will be lost.

`1541SUB` creates a subdirectory which begins at the same location as the directory on a 1541 (track 18, sector 1). A header block for this directory is also created and placed where the header block on a 1541 is located (track 18, sector 0).

`1581SUB` creates a subdirectory which begins at the same location as the directory on a 1581 (track 40, sector 3). A header block for this directory is also created and placed where the header block on a 1581 is located (track 40, sector 0). Due to the way Native Mode subdirectories work, 1581SUB must create two subdirectories to provide the proper headers and directory space. This program has been tested for use with Superbase and Superbase 128 and allows these programs to access larger storage areas.

To use either of these programs, LOAD and RUN the one which you wish to use. The program will ask for the device number of RAMLink as well as the partition number you wish to create the subdirectory in. Do not use either of these programs on a partition which contains useful data, as the partition will be formatted by the utility. You must also make sure that the partition you select for use has enough tracks to support the subdirectory (minumum 18 tracks for 1541SUB, 40 tracks for 1581SUB).

AUTOFILE EDITOR

This utility allow you to create and edit special configuration data stored on RAMLink. By using this program, you can configure RAMLink to automatically load any file you wish from any drive on your system each time the computer is turned on or reset. Two separate areas exist for this purpose - one for a C64 or C128 in 64 mode, and another area for a C128 in 128 mode. This allows 128 users to have separate boot files for each mode on their machines. There are a total of six parameters which may be set for each of these areas. These are:

1. Enable / Disable Autofile loading
2. BASIC or Machine Language file
3. SYS Address if program is Machine Language
4. Device Number of drive where the file is located
5. Memory Bank for load and SYS if file is Machine Language (128 only)
6. Path and Filename of file to be loaded

The program is menu driven, and will only present the options available according to your current mode and settings. It is wise to leave the autofile option disabled until you have properly set the other parameters.

AUTO-BOOT 128

This Public Domain utility has been included to allow you to create boot sectors for use in 128 mode. A boot sector may be created in any Emulation Mode or Native Mode partition. Native Mode partitions automatically protect the boot sector keeping it allocated at all times. Therefore, when using this program on a Native Mode partition, you will be told that the sector is already being used. If you continue with the process at this point, the program will go ahead and create the boot sector.

Before running this program, use the Change Partition command to make the partition that you wish to place the boot sector in the current partition. You may then specify drive 0 when asked for this information within the program.

DISK CRACKER HD

This utility is a modification of a program released to the public domain by its author, Mike Henry. DISK CRACKER HD is a disk editor and monitor intended for use only by those familiar with programs of this type. We will not make any attempt to document this software, since it should only be used by those familiar with the data storage methods used on Commodore compatible disk drives, and these individuals are usually well versed in the use of this type of software.

HARDWARE TEST

This program is used to test the compatibility of your particular hardware with the standard RAMLink unit. It is entirely possible that RAMLink will not operate with some computers due to timing inconsistancies caused by wide variations in the manufacture of Commodore computers. If you notice inconsistant operation with RAMLink (i.e., computer lockup, reset or bombing out of programs) which are not attributable to normal software incompatibility, you should run this test program. Follow the directions given in the program itself after loading and running it. You should allow this test to operate for several hours, starting after your computer has been turned off for several hours. This will allow the program to test your computer at all operating extremes. If this test fails, contact CMD for information on how to adapt your system for use with RAMLink.

RAM TEST

This program will test all RAM attached to RAMLink. Whenever you add new RAM to the system, or suspect problems with the RAM currently installed, load and run this program. Follow the on-screen prompts. Please note that this test is a destructive test, and you will have to reconfigure your system after running it.

ZAP SYSTEM

This program will allow you to completely destroy all configuration and partitioning information stored in RAMLink. This will destroy any data you may have stored on RAMLink, so use it only if you do not wish to retain that data on RAMLink.

REWRITE DOS(.64/.128)

These programs are used only by owners of the CMD HD Series hard drives who wish to use RAMLink and the HD with the parallel cable. Both programs do the same thing, but one runs in 64 mode, while the other runs in 128 mode. You only need to use one of these two porgrams to perform the DOS upgrade. REWRITE DOS will not destroy or alter any of your partitions or the data stored in them - it simply replaces the existing version of HD DOS. If you have RAMLink attached to your system with a parallel cable, issue the Parallel Off (@P0) command before placing the HD into CONFIGURATION Mode. Follow the steps given below to rewrite the DOS on your HD:

1. Place the RAMLink Utilities disk into your floppy disk drive. If you are using a non-JiffyDOS equipped system, this should be device number 8. If you are using JiffyDOS, make sure that the floppy disk drive is your default drive (use the '@#' or <CONTROL>+<D> commands).

2 LOAD and RUN the program called REWRITE DOS.64 or REWRITE DOS.128, depending on which machine (or mode on the 128) you are using.

```
LOAD"REWRITE DOS.64",8          (C64 without JiffyDOS)
RUN

RUN"REWRITE DOS.128"            (C128 without JiffyDOS)

↑"REWRITE DOS.64"               (C64 with JiffyDOS)

↑"REWRITE DOS.128"              (C128 with JiffyDOS)
```

3. Place the HD into CONFIGURATION MODE as instructed on the screen and press RETURN. The new operating system will be written to the system partition on the HD.

4. Press RESET on the HD when finished.

> Note: REWRITE DOS should only be loaded and run from a floppy disk drive. Other programs on the floppy disk will be needed while the program is running, so do not remove the disk during this process. NEVER press the HD's RESET button while writing a new DOS.

Appendix B
Error Codes

The error codes used by RAMLink have been arranged to be compatible with the codes used on Commodore floppy disk drive units. Some errors have been eliminated on RAMLink since certain conditions that exist in floppy drives do not occur in the same manner on a RAM drive. Whenever errors are encountered on your RAMLink, these codes should help in localizing the problem.

Errors are returned over the command channel in the following format:

```
ec,estring,tv,sv
```

where:
- ec = a two-digit error number
- estring = an ASCII string describing the error
- tv = the track variable (the logical track where the error took place)
- sv = the sector variable (the logical sector where the error took place)

Note: Some errors define special meanings for the track and sector variables and are described below when necessary.

Command Channel Error Codes

0 0 **OK** (not an error)
This code is present when no other error condition exists.

0 1 **FILES SCRATCHED** (not an error)
Occurs after using the DOS or BASIC scratch commands. The number of files scratched will be indicated in the track variable, and the sector variable will contain a zero.

0 2 **PARTITION SELECTED** (not an error)
Occurs after switching partitions with the RL DOS 'CP' command, and after changing 1581 sub-partitions with the '/' command. After using the 'CP' command, the track variable contains the partition number of the newly selected partition, while the sector variable contains zero. After the '/' command is issued, the track variable contains the number of the first track in the 1581 sub-partition, while the sector variable contains the number of the last track in the 1581 sub-partition.

26 **WRITE PROTECT ON**
Indicates that an attempt was made to write to a RAMLink partition while that partition was write protected.

30 **SYNTAX ERROR** (general)
Indicates that the RL DOS command interpreter was unable to identify the last command sent via the command channel. This is usually caused by incorrect characters being present in the disk command.

31 **SYNTAX ERROR** (unrecognized command)
Usually indicates that the first character of the last command string was not recognized as part of a legal DOS command.

32 **SYNTAX ERROR** (command string too long)
Occurs when a disk command string contains over 254 characters. Due to the large size of the command channel input buffer in RAMLink, this error should rarely be encountered.

33 **SYNTAX ERROR** (illegal file name)
Usually indicates that an attempt was made to use wildcards or pattern matching within a file name or disk command that does not accept wildcards.

34 **SYNTAX ERROR** (missing file name)
The last disk command failed due to a missing file name or the file name should have been preceded by a colon (:).

48 **ILLEGAL JOB** (in job queue)
The last job code placed into the job queue was not legal.

50 **RECORD NOT PRESENT**
The last attempt to access a relative (REL) file record specified a record number which does not yet exist. This condition will result even when attempting to create or expand a relative file, and under those conditions should be ignored.

51 **OVERFLOW IN RECORD**
The last attempted write to a relative record contained more data than could be stored in the record. When this occurs, as much data as can fit is stored into the record.

52 **FILE TOO LARGE**
The last attempt to access a relative record would have exceeded the amount of storage space remaining in the partition specified. Under this condition, no additional records were added to the file, and if this was an attempt to create a new relative file, the file was not created.

60 **WRITE FILE OPEN**
The last attempt to open a file was made to a file which was already open for writing.

61 **FILE NOT OPEN**
Indicates that the last attempt to access data was made to a file which was not properly opened. Under some circumstances, this error is not generated and the attempt to access the file is simply ignored.

62 **FILE NOT FOUND**
During the last attempt to open a file, the DOS could not find the file specified by the path and filename given. This error will also occur if the filetype of the file does not match the allowed filetypes.

63 **FILE EXISTS**
While attempting to open a new file, another file with the same name was found in the specified partition or sub-directory.

64 **FILE TYPE MISMATCH**
The last disk operation specified a file which did not match the filetypes allowed for that operation.

65 **NO BLOCK**
Indicates that an attempt was made to allocate a block which was already allocated using the BLOCK-ALLOCATE (B-A) command. The track and sector variables will contain the next available block when this condition occurs. If the track variable contains a zero, there are no blocks available.

66 **ILLEGAL BLOCK**
Occurs when an attempt is made to access a track or sector which does not exist. This may occur if one of the track and sector links within a file have become corrupted.

67 **ILLEGAL BLOCK**
Usually caused by a corrupt disk. This error shows up when the block parameters for a partition do not exist.

70 **NO CHANNEL**
This indicates that the specified channel within a disk command is already in use, or that all buffers in the drive are currently in use. This may be an indication that too many files are currently opened.

71 **DIRECTORY ERROR**
Indicates that the BAM (Block Availability Map) on the diskette is invalid. To correct the problem, Validate the disk.

7 2 **PARTITION FULL**
Occurs when the targeted partition or its directory are full. When this message is sent, there are still two blocks free in the partition, allowing the current file to be closed.

7 4 **DRIVE NOT READY**
Occurs after an attempt was made to access an illegal partition or a partition that has not been properly formatted.

7 7 **SELECTED PARTITION ILLEGAL**
Occurs when an attempt has been made to enter a nonexistent (or illegal) partition.

Appendix C
Partition and File Formats

When accessing individual tracks & sectors on RAMLink, it is imporrtant to remember that the track and sector layout is specific to the type of partition in which the access occurs. It is through this method that RAMLink is able to attain its high level of compatibility. Therefore, when accessing an emulation mode partition, the track and sector layout of the partition is identical to that of the drive that it emulates. The tables in this section should be used as a guide to indicate which tracks and sectors are available in each type of partition, and where the header, BAM and directory blocks are located. This appendix also provides information about the format of BAM and directory entries, as well as the format of different file types.

Common Formats Used in all Partition Types

DIRECTORY FILE TABLE FORMAT (ALL PARTITION TYPES) 1541 & 1571 PARTITIONS - TRACK 18 SECTOR 1 1581 PARTITION - TRACK 40 SECTOR 3 NATIVE PARTITION - TRACK 1 SECTOR 34	
BYTE	**DESCRIPTION**
0	Track pointer to next directory block (0 indicates last block)
1	Sector pointer to next directory block (255 indicates last block)
2 - 31	File entry 1 (see Figure C2)
32 - 33	Two zero (0) bytes (reserved)
34 - 63	File entry 2 (see Figure C2)
64 - 65	Two zero (0) bytes (reserved)
66 - 95	File entry 3 (see Figure C2)
96 - 97	Two zero (0) bytes (reserved)
98 - 127	File entry 4 (see Figure C2)
128 - 129	Two zero (0) bytes (reserved)
130 - 159	File entry 5 (see Figure C2)
160 - 161	Two zero (0) bytes (reserved)
162 - 191	File entry 6 (see Figure C2)
192 - 193	Two zero (0) bytes (reserved)
194 - 223	File entry 7 (see Figure C2)
224 - 225	Two zero (0) bytes (reserved)
226 - 255	File entry 8 (see Figure C2)

Figure C1

DIRECTORY FILE ENTRY FORMAT ALL PARTITION TYPES AND NATIVE MODE SUBDIRECTORIES		
BYTE	**VALUE**	**DESCRIPTION**
0		File type:
	0	DEL (Deleted)
	1	SEQ (Sequential
	2	PRG (Program)
	3	USR (User)
	4	REL (Relative)
	5	CBM (1581 style sub-partition)
	6	DIR (Native Mode subdirectory)
		Note: Filetypes will be OR'ed with $80 when the file has been properly closed. Filetypes will be OR'ed with $C0 if the file is locked.
1		Track pointer to first data block (or header block if filetype is DIR)
2		Sector pointer to first data block (or header block if filetype is DIR)
3 - 18		Filename padded with shifted spaces ($A0)
19		Pointer to starting track of side sector or super side sector if filetype is REL
20		Pointer to starting sector of side sector or super side sector if filetype is REL
21		Record length if filetype is REL
22	0	Reserved
23		Year file was created (last two digits)
24		Month file was created
25		Day file was created
26		Hour file was created
27		Minute file was created
28		Number of blocks used by file (low byte)
29		Number of blocks used by file (high byte)

Figure C2

1541 and 1571 Emulation Mode Partitions

SECTORS PER TRACK (1541 EMULATION MODES)		
TRACK RANGE	SECTORS AVAILABLE	TOTAL
1 through 17	0 through 20	21
18 through 24	0 through 18	19
25 through 30	0 through 17	18
31 through 35	0 through 16	17

Figure C3

SECTORS PER TRACK (1571 EMULATION MODE)		
TRACK RANGE	SECTORS AVAILABLE	TOTAL
1 through 17	0 through 20	21
18 through 24	0 through 18	19
25 through 30	0 through 17	18
31 through 35	0 through 16	17
36 through 52	0 through 20	21
53 through 59	0 through 18	19
60 through 65	0 through 17	18
66 through 70	0 through 16	17

Figure C4

HEADER & BAM (1541 & 1571 EMULATION MODES) TRACK 18 SECTOR 0		
BYTE	VALUE	DESCRIPTION
0	18	Track pointer to first directory block
1	1	Sector pointer to first directory block
2	65	ASCII 'A' for format type
3	0 128	1541 Emulation Mode 1571 Emulation Mode
4 - 143		BAM (Block Availability Map)
144 - 161		Disk name padded with shifted spaces
162 - 163		Disk ID
164	160	Shifted Space for separator
165	50	ASCII '2' for DOS version
166	65	ASCII 'A' for format type
167 - 170	160	Shifted spaces for separators
171 - 220	0	Null bytes - reserved
221 - 255	0	1541 Emulation mode - Null bytes - reserved 1571 Emulation mode - Number of sectors available for tracks 36 through 70 - one byte per track (part of 1571 (side 2) BAM)

Figure C5

BAM for 1571 (side 2) EMULATION MODE TRACK 53 SECTOR 0		
BYTE	**VALUE**	**DESCRIPTION**
0 - 104		BAM for tracks 36 through 70 (3 bytes per track)
105 - 255	0	Null bytes - reserved

Figure C6

BAM ENTRY FORMAT 1541 & 1571 (side 1) EMULATION MODES Format of bytes 4-143 in Track 18 Sector 0 (Figure C5) 4 bytes per track: bytes 4-7 cover track 1, bytes 8-11 cover track 2, ...	
BYTE	**DESCRIPTION**
0	Number of sectors available on track
1	Block Availability for sectors 0 - 7
2	Block Availability for sectors 8 - 15
3	Block Availability for sectors 16 - 23
Notes: The lowest bit (LSB) in each byte (bytes 1 through 3) indicates the status of the lowest sector covered by that byte. A binary value of 1 indicates that the sector is available, while a value of 0 indicates that the sector is allocated.	

Figure C7

BAM ENTRY FORMAT 1571 (side 2) EMULATION MODE Format of bytes 0 - 104 on Track 53 Sector 0 (Figure C6) 3 bytes per track: bytes 0-2 cover track 36, bytes 3-5 cover track 37, ...	
BYTE	**DESCRIPTION**
0	Block Availability for sectors 0 - 7
1	Block Availability for sectors 8 - 15
2	Block Availability for sectors 16 - 23
Notes: The lowest bit (LSB) in each byte (bytes 0 through 2) is used to indicate the status of the lowest sector covered by that byte. A binary value of 1 indicates that the sector is available, while a value of 0 indicates that the sector is allocated. The associated byte for the number of sectors available on each track is stored in bytes 221 through 225 of track 18 sector 0 (see Figure C5).	

Figure C8

1581 Emulation Made Partitions

SECTORS PER TRACK		
TRACK RANGE	SECTORS AVAILABLE	TOTAL
1 through 80	0 through 39	40

Figure C9

DIRECTORY HEADER TRACK 40 SECTOR 0		
BYTE	VALUE	DESCRIPTION
0	40	Track pointer to first directory block
1	3	Sector pointer to first directory block
2	68	ASCII 'D' for format type
3	0	Reserved
4 - 21		Disk name padded with shifted spaces
22 - 23		Disk ID
24	160	Shifted Space for separator
25	51	ASCII '3' for DOS version
26	68	ASCII 'D' for format type
27 - 28	160	Shifted spaces for separators
29 - 255	0	Null bytes - reserved

Figure C10

BAM BLOCK1 TRACK 40 SECTOR 1		
BYTE	VALUE	DESCRIPTION
0	40	Track pointer to next BAM block
1	2	Sector pointer to next BAM block
2	68	ASCII 'D' for DOS version
3	187	Complement of version number
4 - 5		Disk ID
6	192	Not used in RAMLink - set to 1581 default value
7	0	Flag for Auto Loader file
8 - 15	0	Reserved
16 - 255		BAM for tracks 1 through 40 (6 bytes per track)

Figure C11

BAM 2 TRACK 40 SECTOR 2		
BYTE	**VALUE**	**DESCRIPTION**
0	0	Indicates last sector for BAM
1	255	Indicates all bytes in sector used
2	68	ASCII 'D' for DOS version (copy)
3	187	Complement of version number (copy)
4 - 5		Disk ID (copy)
6	192	Not used in RAMLink - set at 1581 default value
7	0	Flag for Auto Loader file (copy)
8 - 15	0	Reserved
16 - 255		BAM for tracks 41 through 80 (6 bytes per track)

Figure C12

BAM ENTRY FORMAT Format of bytes 16 - 255 in Track 40 Sectors 1 and 2 (Figures C11 & C12)	
BYTE	**DESCRIPTION**
0	Number of sectors available on track
1	Block Availability for sectors 0 - 7
2	Block Availability for sectors 8 - 15
3	Block Availability for sectors 16 - 23
4	Block Availability for sectors 24 - 31
5	Block Availability for sectors 32 - 39
Notes: The lowest bit (LSB) in each byte (bytes 1 through 5) indicates the status of the lowest sector covered by that byte. A binary value of 1 indicates that the sector is available, while a value of 0 indicates that the sector is allocated.	

Figure C13

Native Made Partitions

SECTORS PER TRACK		
TRACK RANGE	SECTORS AVAILABLE	TOTAL
1 through 255	0 through 255	256

Figure C14

ROOT DIRECTORY AND SUBDIRECTORY HEADER TRACK 1 SECTOR 1 FOR ROOT DIRECTORY VARIES FOR SUBDIRECTORIES		
BYTE	VALUE	DESCRIPTION
0		Track pointer to first directory block
1		Sector pointer to first directory block
2	72	ASCII 'H' for format type
3	0	Reserved
4 - 21		Disk name padded with shifted spaces
22 - 23		Disk ID
24	160	Shifted space for separator
25	49	ASCII '1' for DOS version
26	72	ASCII 'H' for format type
27 - 28	160	Shifted spaces for separators
29 - 31	0	Reserved
32	1	Pointer to ROOT header track
33	1	Pointer to ROOT header sector
34		Track pointer to DIR PARENT header
35		Sector pointer to DIR PARENT header
36		Track pointer to DIR entry in PARENT directory
37		Sector pointer to DIR entry in PARENT directory
38		Index to starting byte of DIR entry in PARENT directory
39 - 255	0	Null bytes - reserved

Figure C15

NATIVE MODE BAM (1st BAM block) TRACK 1 SECTOR 2		
BYTE	**VALUE**	**DESCRIPTION**
0	0	Reserved
1	0	Reserved
2	72	ASCII 'H' for format type
3	183	Complement of format type
4 - 5		Disk ID
6	192	Not used in RAMLink - set at 1581 default value
7	0	Flag for Auto Loader file
8		Track number of last available track in partition
9 - 31	0	Reserved
32 - 255	0	BAM for tracks 1 through 7 (32 bytes per track)

Figure C16

NATIVE MODE BAM (blocks 2-32) TRACK 1 SECTORS 3 - 28		
TRACK	**SECTOR**	**DESCRIPTION**
1	3	BAM for tracks 8 through 15 (32 bytes per track)
1	4	BAM for tracks 16 through 23 (32 bytes per track)
1	5	BAM for tracks 24 through 31 (32 bytes per track)
1	6	BAM for tracks 32 through 39 (32 bytes per track)
1	7	BAM for tracks 40 through 47 (32 bytes per track)
1	8	BAM for tracks 48 through 55 (32 bytes per track)
1	9	BAM for tracks 56 through 63 (32 bytes per track)
1	10	BAM for tracks 64 through 71 (32 bytes per track)
1	11	BAM for tracks 72 through 79 (32 bytes per track)
1	12	BAM for tracks 80 through 87 (32 bytes per track)
1	13	BAM for tracks 88 through 95 (32 bytes per track)
1	14	BAM for tracks 96 through 103 (32 bytes per track)
1	15	BAM for tracks 104 through 111 (32 bytes per track)
1	16	BAM for tracks 112 through 119 (32 bytes per track)
1	17	BAM for tracks 120 through 127 (32 bytes per track)
1	18	BAM for tracks 128 through 135 (32 bytes per track)
1	19	BAM for tracks 136 through 143 (32 bytes per track)
1	20	BAM for tracks 144 through 151 (32 bytes per track)
1	21	BAM for tracks 152 through 159 (32 bytes per track)
1	22	BAM for tracks 160 through 167 (32 bytes per track)
1	23	BAM for tracks 168 through 175 (32 bytes per track)
1	24	BAM for tracks 176 through 183 (32 bytes per track)
1	25	BAM for tracks 184 through 191 (32 bytes per track)
1	26	BAM for tracks 192 through 199 (32 bytes per track)
1	27	BAM for tracks 200 through 207 (32 bytes per track)
1	28	BAM for tracks 208 through 215 (32 bytes per track)
1	29	BAM for tracks 216 through 223 (32 bytes per track)
1	30	BAM for tracks 224 through 231 (32 bytes per track)
1	31	BAM for tracks 232 through 239 (32 bytes per track)
1	32	BAM for tracks 240 through 247 (32 bytes per track)
1	33	BAM for tracks 248 through 255 (32 bytes per track)

Figure C17

NATIVE MODE BAM ENTRY FORMAT Format of bytes 32 - 255 in Track 1 Sector 2 and bytes 0 - 255 inTrack 1 Sectors 3 - 33 (Figures C11 & C12)	
BYTE	**DESCRIPTION**
0	Block Availability for sectors 0 - 7
1	Block Availability for sectors 8 - 15
2	Block Availability for sectors 16 - 23
3	Block Availability for sectors 24 - 31
4	Block Availability for sectors 32 - 39
5	Block Availability for sectors 40 - 47
6	Block Availability for sectors 48 - 55
7	Block Availability for sectors 56 - 63
8	Block Availability for sectors 64 - 71
9	Block Availability for sectors 72 - 79
10	Block Availability for sectors 80 - 87
11	Block Availability for sectors 88 - 95
12	Block Availability for sectors 96 - 103
13	Block Availability for sectors 104 - 111
14	Block Availability for sectors 112 - 119
15	Block Availability for sectors 120 - 127
16	Block Availability for sectors 128 - 135
17	Block Availability for sectors 136 - 143
18	Block Availability for sectors 144 - 151
19	Block Availability for sectors 152 - 159
20	Block Availability for sectors 160 - 167
21	Block Availability for sectors 168 - 175
22	Block Availability for sectors 176 - 183
23	Block Availability for sectors 184 - 191
24	Block Availability for sectors 192 - 199
25	Block Availability for sectors 200 - 207
26	Block Availability for sectors 208 - 215
27	Block Availability for sectors 216 - 223
28	Block Availability for sectors 224 - 231
29	Block Availability for sectors 232 - 239
30	Block Availability for sectors 240 - 247
31	Block Availability for sectors 248 - 255
Notes: The lowest bit (LSB) in each byte (bytes 0 through 31) indicates the status of the highest sector covered by that byte. A binary value of 1 indicates that the sector is available, while a value of 0 indicates that the sector is allocated.	

Figure C18

File Formats

PROGRAM FILE FORMAT	
BYTE	**DESCRIPTION**
0	Pointer to track of next file block (contains a zero if current block is last block in file).
1	Pointer to sector of next file block (contains pointer to last byte used if current block is last block in file).
2 - 255	Program data (bytes 2 and 3 contain load address of program in low byte-high byte format if current block is first block in file).

Figure C19

SEQUENTIAL FILE FORMAT	
BYTE	**DESCRIPTION**
0	Pointer to track of next file block (contains a zero if current block is last block in file).
1	Pointer to sector of next file block (contains pointer to last byte used if current block is last block in file).
2 - 255	Data bytes .

Figure C20

RELATIVE FILE DATA BLOCK FORMAT	
BYTE	**DESCRIPTION**
0	Pointer to track of next data file block (contains a zero if current block is last data block in file).
1	Pointer to sector of next data file block (contains pointer to last byte used if current block is last data block in file).
2 - 255	Data bytes .Empty records will begin with a $FF in the first byte of the record, the remaining bytes will contain $00 bytes. Partially filled records will also contain $00 bytes in the unused portion of the record.

Figure C21

RELATIVE FILE SUPER SIDE SECTOR BLOCK FORMAT	
BYTE	**DESCRIPTION**
0	Pointer to track of first side sector in first group (group 0).
1	Pointer to sector of first side sector in first group (group 0).
2	Super side sector identification byte ($FE)
3 - 254	Track and sector pointers to first side sector of 126 groups (groups 0 through 125, two bytes per pointer). Unused group pointers contain $00 bytes.

Figure C22

RELATIVE FILE SIDE SECTOR BLOCK FORMAT	
BYTE	**DESCRIPTION**
0	Pointer to track of next side sector in this group (contains a zero if current block is last side sector block in use).
1	Pointer to sector of next side sector in this group (contains pointer to last byte used if current block is last side sector block in use).
2	Side sector number (0 - 5)
3	Record length of associated relative file
4	Pointer to track of first side sector in this group (number 0).
5	Pointer to sector of first side sector in this group (number 0).
6 - 7	Pointer to track and sector of second side sector in this group (number 1).
8 - 9	Pointer to track and sector of third side sector in this group (number 2).
10 - 11	Pointer to track and sector of fourth side sector in this group (number 3).
12 - 13	Pointer to track and sector of fifth side sector in this group (number 4).
14 - 15	Pointer to track and sector of sixth side sector in this group (number 5).
16 - 255	Track and sector pointers to 120 data blocks (two bytes per pointer). Unused data block pointers contain $00 bytes.

Figure C23

Appendix D
RAMLink Memory Map

RAMLink Partitionable RAM

RAMLink actually contains two separate memory maps. One is used by RL DOS for keeping track of important system variables, as well as for handling the emulation of many disk drive fuctions and memory locations (sich as the Job Queue buffer). The second map contains all the partitionable RAM. A map of this area is given below.

The SYSTEM partition is always located at the very top of any available memory.

There is almost always some portion of the memory which cannot be partitioned.

Memory which is partitioned starts at the very bottom of available memory (memory address $0000). Bear in mind that the memory map used for memory is an alternate map which only contains partitionable RAM. Due to this method of handling the RAM, performing a MEMORY-READ to location $0000 of RAMLink looks at a different area (DOS memory) than performing a Direct RAM Access read of location $0000 (which would look into the partionable RAM).

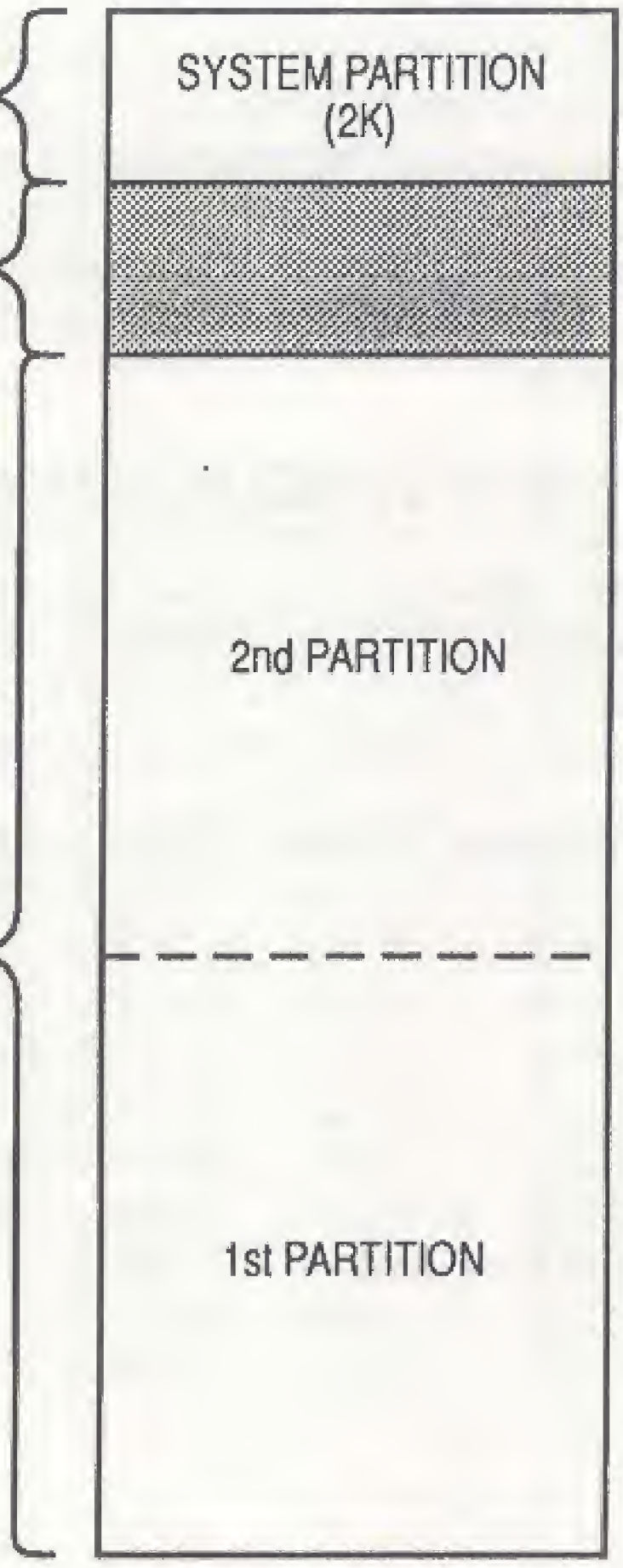


Figure D1

How Partitions are Allocated

Partitions are created beginning at the very bottom of partitionable memory. Partition numbers do not always correlate with a partitions actual placement. Instead, the order in which partitions are created determines where the partitions will be located in partitionable memory. For example, if you create partition number 1, then partition number 5, and finally partition number 2, then partition 1 will be the lowest in memory, partition 2 will be the highest in memory, with partition 5 existing between the two.

Deleting partitions will change this arrangement to some degree. Using the example just given, if you delete partition 5, then partition 2 will be moved so that it now begins where partition 5 used to be. This has the effect of filling up gaps in the partitionable memory, leaving you the maximum amount of RAM available for more partitions.

Since partitions do have certain size restrictions, there will amost always be some amount of RAM which will not be partitionable. Careful planning of your paritions can reduce this to a minimum. When planning partitions, it is usually wise to keep any Foreign (Direct Access) partitions at the very bottom of the memory map, so create these first. When figuring partition sizes it may help to know that there are 4096 Commodore blocks in 1 Megabyte.

Useful RAMLink Memory Locations

The following chart gives information on a number of useful memory locations within the RAMLink DOS area. Bear in mind that this area is separate from the area described above. Do not confuse the DOS RAM area with the SYSTEM partition - these are separate entities.

Address Range	Description of Contents
$0000 - $0004	1541/1571 Emulation Mode Job Queue
$0006 - $000F	1541/1571 Job Queue Track & Sector Variables
$0002 - $000A	1581 Emulation Mode Job Queue
$000B - $001C	1581 Job Queue Track & Sector Variables
$0020 - $0027	Native Mode Job Queue
$0028 - $002F	Native Mode Job Queue Track & Sector Variables
$0030 - $00FF	Emulated Zero Page Variables
$0100 - $01FF	Variables
$0200 - $027F	Input Command Buffer
$0280 - $02FF	Variables

Appendix E
Installing RAMCard

Expanding the memory in RAMLink requires a 'daughter board' called RAMCard. RAMLink may be purchased with or without this board. If you purchased RAMLink without a RAMCard, you may purchase one at a later date from CMD. This section will instruct you on how to install RAMCard into your RAMLink. Follow the steps given below to accomplish the installation.

1. Turn off your computer and remove ALL power from RAMLink (including a backup battery if you have one).

2. Remove any devices you may have plugged into RAMLink, including RAM expanders, cartridges and the parallel cable.

3. Disconnect RAMLink from your computer.

4. Using an Allen wrench, remove the four screws which hold the two halves of the RAMLink case together.

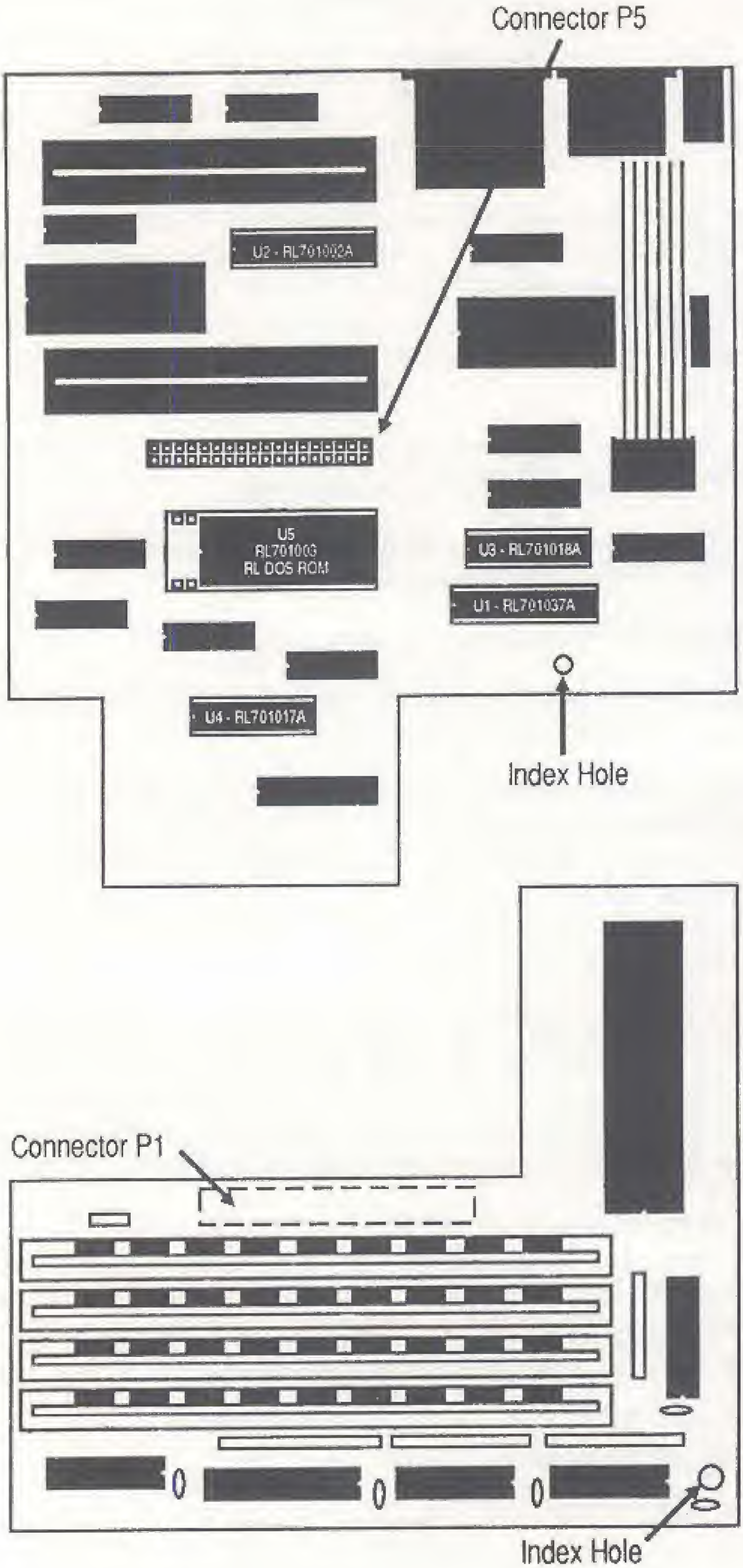
5. Carefully slide the top half of the RAMLink case off of the bottom half. This is most easily accomplished by prying the case open at the mouth of the connector that attaches to the computer, swinging the top up and away from the bottom. Use caution in doing this, as there is a ribbon cable inside the case which connects a board located in the top half to the main board loacated in the bottom half.

6. Locate the plastic stand-off which comes with RAMCard, and press the top end (opposite the end with the wings) into the index hole in RAMCard, inserting it from the bottom side of RAMCard. The bottom side is the side without components, and the index hole is located in the bottom right corner of RAMCard as shown in figure on the next page.

7. Line up the plastic stand-off and Connector P1 on RAMCard with the index hole and Connector P5 on RAMLink's main board. Press Connector P1 firmly into place on Connector P5. Double-check to make sure that these connectors are mated correctly, then press down on the plastic stand-off until it seats fully on RAMLink's main board.

8. Install any SIMMs you wish to have on RAMCard if you have any, or have not already done so.

9. Re-assemble RAMLink by putting the two halves of the cover back together, and fasten the case halves back together with the screws.


Connector P5
U2 - RL701002A
U5
RL701003
RL DOS ROM
U3 - RL701018A
U1 - RL701037A
U4 - RL701017A
Index Hole
Connector P1
Index Hole

Appendix F
Installing SIMMs

Expanding the memory in RAMLink can be performed by adding SIMM modules to RAMCard (provided you have a RAMCard in your RAMLink). RAMCard contains four sockets for holding SIMMs. To add SIMMs to RAMCard, you must have at least one of these sockets open. RAMCard is capable of holding two different capacity SIMMs. The lower capacity SIMM is organized as 1 Megabits x 8 (a 1 Megabyte SIMM). The higher capacity SIMM is organized as 4 Megabits x 8 (a 4 Megabyte SIMM). Since RAMCard has a total of four sockets, using the lower capacity SIMMs gives a maximum of 4 Megabytes of RAM, while using the higher capacity SIMMs allows a maximum of 16 Megabytes of RAM. Mixing of the two capacity SIMMs is not allowed, and to use the higher capacity SIMMs you must cut a trace on RAMCard as shown in the diagram below. SIMMs must also be mounted in a particular order, starting with the socket marked SIMM 0, followed by SIMM 1, SIMM 2, and finally SIMM 3. Be careful when pulling old SIMMs out, as there are retainers at each end of the sockets which hold the SIMMs tightly in place. Also, be sure to take appropriate precautions to avoid static electricity, which could damage your SIMMs. SIMMs must be mounted with the chip side of the modules facing the rear of RAMLink. SIMM speed must be 100ns or faster.

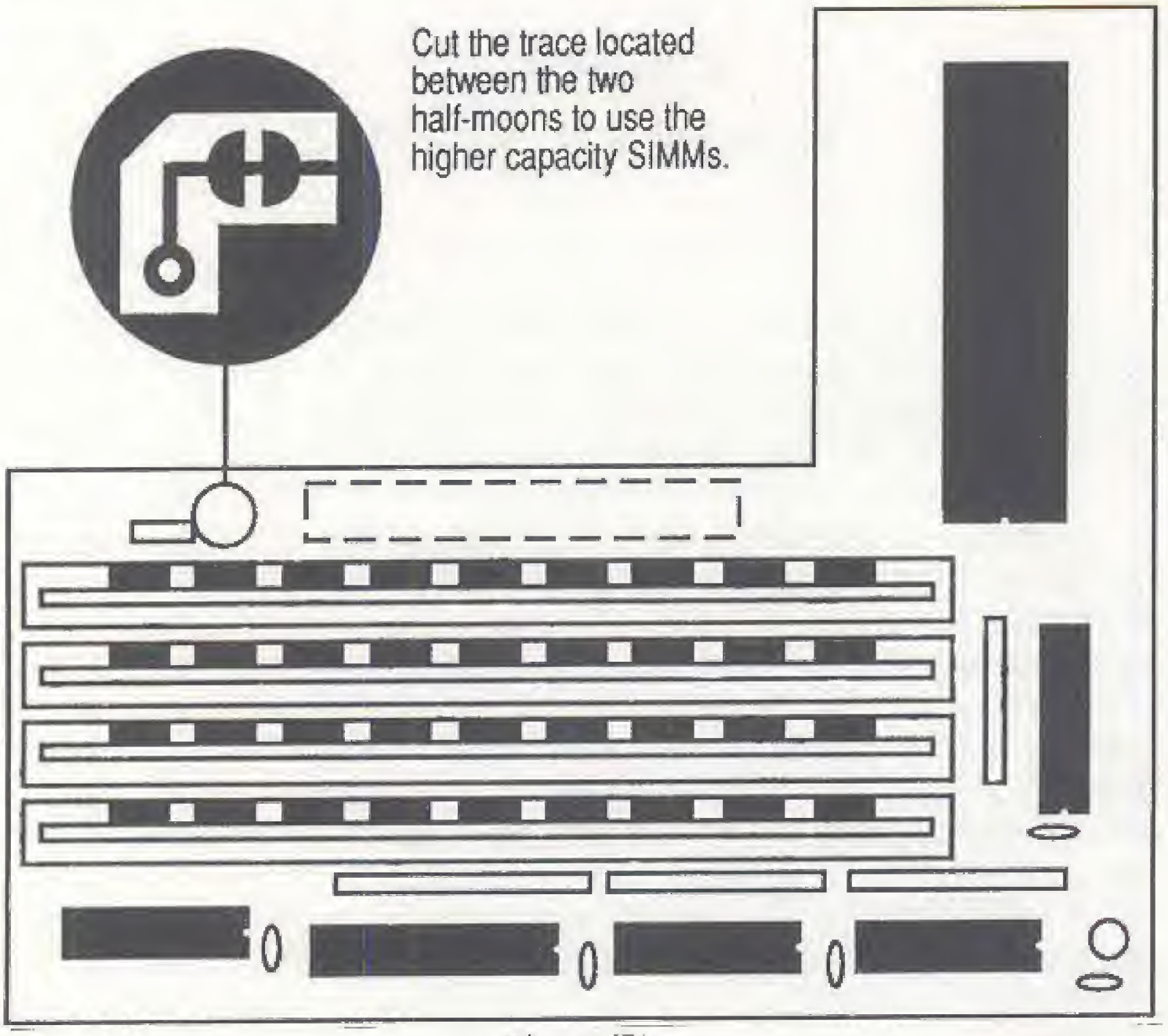

Cut the trace located between the two half-moons to use the higher capacity SIMMs.

Figure F1

Appendix G
Parallel Port

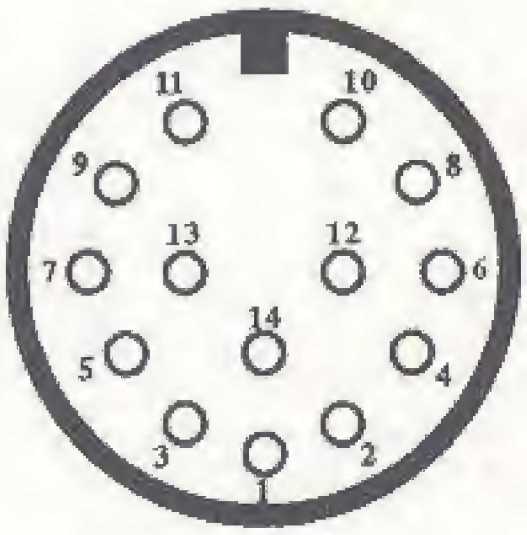


Figure G1

PIN NO.	PIN NAME	DESCRIPTION
1	$\overline{\text{PCLK}}$	PARALLEL CLOCK
2	GND	GROUND
3	$\overline{\text{PEXT}}$	RESERVED
4	PD1	PARALLEL DATA BIT 1
5	PD4	PARALLEL DATA BIT 4
6	GND	GROUND
7	$\overline{\text{PRDY}}$	PARALLEL READY
8	PD0	PARALLEL DATA BIT 0
9	PD7	PARALLEL DATA BIT 7
10	$\overline{\text{PATN}}$	PARALLEL ATTENTION
11	PD5	PARALLEL DATA BIT 5
12	PD3	PARALLEL DATA BIT 3
13	PD6	PARALLEL DATA BIT 6
14	PD2	PARALLEL DATA BIT 2

Figure G2

Appendix H
Power Connector

Figure H1

PIN NO.	DESCRIPTION
1	+5 VDC (Requires 1 Ampere minimum)
2	GROUND
3	GROUND
4	+12 VDC (Requires .5 Ampere minimum)

Figure H2

Appendix I
Battery Connector

Figure I1

The battery connector is intended to mate with a sealed lead-acid type which supplies 6 volts DC at 6.5 Ampere-hours. RAMLink provides a trickle-charge circuit in order to keep the battery fully charged. Connector polarity is positive tip.

LIMITED WARRANTY

Creative Micro Designs, Inc., 50 Industrial Dr., P.O. Box 646, East Longmeadow, Massachusetts warrants to the original retail purchaser of the RAMLink that it is free of defects in material and workmanship for a period of 90 days from date of purchase from an authorized CMD dealer or 90 days from the date of delivery if purchased direct from CMD.

IMPLIED WARRANTIES, OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR OTHERWISE, ARE LIMITED IN DURATION TO THE DURATION OF THE EXPRESS WARRANTY SET FORTH ABOVE. IN NO EVENT SHALL CMD BE LIABLE FOR ANY LOSS, INCONVENIENCE, OR DAMAGE WHETHER DIRECT, INCIDENTAL, CONSEQUENTIAL OR OTHERWISE RESULTING FROM BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR OTHERWISE, WITH RESPECT TO THE EQUIPMENT, EXCEPT AS SET FORTH HEREIN.

SOME STATES DO NOT ALLOW THE LIMITATIONS ON THE LIFE OF AN IMPLIED WARRANTY. SOME STATES MAY ALSO DISALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT PERTAIN TO YOU.

DESCRIPTION OF WARRANTY RIGHTS

From the date of purchase or date of delivery, in the case of a direct sale through CMD, through the warranty period, CMD will, at its discretion, repair or replace any part deemed to be defective at no charge for parts/labor to the original retail customer. During the applicable warranty period wherein CMD will repair or replace defective parts without charge for labor, all warranty inspections and repairs must be performed at a CMD authorized service agency or by CMD itself.

CONDITIONS TO WARRANTY SERVICE

For this warranty to become effective the following requirements must be met:

1. Any postage, insurance and shipping charges of warranted items to a CMD authorized service agency or CMD itself must be prepaid by the original retail purchaser and these costs are not included under this warranty. Return shipping during the warranty period will be paid by CMD to addresses in the continental U.S. All other addresses will be charged for shipping, insurance and any other charges related to the return shipping of the item.
2. The dealer's original bill of sale or a charge or credit or delivery receipt must be retained by the original retail purchaser as proof of purchase date of the warranted item and must be presented to the CMD authorized service agency or CMD itself when warranty claims are advanced.
3. The warranty registration card must be filled out and returned to CMD within 30 days of purchase. If CMD does not receive, in good condition, the warranty registration card within the 30 day period, all warranty services are forfeited by the original retail purchaser.

4. Any CMD product being returned for warranty repairs must be in its original shipping container or one of equivalent structure.

EXCLUSIONS FROM THE WARRANTY

This warranty does not cover the specific items/or conditions described below:

1. Equipment which has been damaged due to:
   - Accident, misuse, abuse, fire, flood, or "Acts of God" or other contingencies beyond the control of CMD.
   - Use of incorrect line voltages.
   - Improper or insufficient ventilation.
   - Failure to follow CMD's operating instructions.
   - Improper or unauthorized repair's.
   - Any unauthorized modification to the device.
   - Improper return packaging or damages caused by failure to insure.
2. Damage to warranted items sustained in shipment to the original retail purchaser.
3. Power transformer voltage or Power Supply conversion to foreign or domestic voltage or current frequency.
4. Any damage resulting from the infection of the unit by a computer virus.
5. Routine adjustments.
6. Damage resulting from the commercial use of this unit.

CMD will not be responsible for labor charges of unauthorized service agencies. CMD will not be responsible for labor charges from CMD authorized service agencies or CMD itself except during the warranty period applicable thereto. CMD will not be responsible for the loss or damage to equipment while in the possession of a CMD authorized service agency. CMD reserves the right to make changes in its design and improvements upon its product without assuming the obligation to install such changes on any of its products previously manufactured.

This warranty gives you specific legal rights and you may also have other rights which vary state to state.

RETURN POLICY

This unit may be returned to Creative Micro Designs, Inc. within 30 days of purchase for a refund of the purchase price less a 10% restocking fee. Shipping charges and taxes are not refundable.

Goods being returned must be returned in original condition in the original shipping container, freight prepaid, and must also include all accessories and be accompanied by a letter stating the reason for return. This letter should contain a return authorization number obtained from Creative Micro Designs, Inc. The return authorization number should also be clearly visable in large characters on the shipping carton.